高等学校教材

计 算 方 法

张世禄　何洪英　编著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书比较全面地介绍了科学与工程计算中常用的计算方法，具体介绍了这些计算方法的基本理论与实际应用，同时对这些数值计算方法的计算效果、稳定性、收敛效果、适用范围以及优劣性与特点也做了简要分析。全书共11章，主要介绍数值代数和数值逼近中常用的实用算法，书中的所有算法都用带计算过程和计算条件的数学语言描述。凡可以手算的算法都附有带计算过程的算例。书中较为详细地介绍了变带宽压缩存储平方根法和压缩存储Seidel迭代法，并附有C程序。书中所有算例的结果都用程序验证过，保证无错，书中有些内容是作者的科研成果。

本书可作为高等院校数学与应用数学、信息与计算科学、应用物理学、计算机科学等专业的高年级本科生和工科硕士研究生使用，也可供从事科学与工程计算的科技工作者参考。

图书在版编目（CIP）数据

计算方法/张世禄，何洪英编著．—北京：电子工业出版社，2010.8
（高等学校教材）
ISBN 978-7-121-11426-7

Ⅰ. ①计…　Ⅱ. ①张… ②何…　Ⅲ. ①计算方法－高等学校－教材　Ⅳ. ①O241

中国版本图书馆 CIP 数据核字（2010）第 140804 号

策划编辑：王赫男
责任编辑：谭海平
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×980　1/16　印张：15　字数：324 千字
印　　次：2010 年 8 月第 1 次印刷
印　　数：4 000 册　　定价：27.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

计算方法是数学中的一个古老分支，自计算机问世之后，计算方法得到了飞速发展．计算方法是计算（computing）学科各专业、数学各专业学生的必修课，也是不少理工专业学生的必修课和选修课．考虑到传统关系，主要考虑到课时关系，计算方法不包含有限单元法、快速 Fourier 变换等算法，也不包含最优逼近、最优化算法，仍只由数值代数和数值逼近组成．

本书的特点如下．

1．全书所有算法都用带计算过程和计算条件的数学语言描述．

计算方法主要是计算机使用的算法，计算机只能直接或间接识别人们用计算机语言编写的程序．程序也是算法，是计算机使用的算法．程序设计就是将非计算机使用的算法翻译成计算机使用的算法．带计算过程和计算条件的数学公式和计算机语言有一对一的映射关系，很容易翻译成计算机语言．

现有计算方法教材中，有少数算法是用自然语言描述的，例如，解 $f(x)=0$ 的对分法，求三角矩阵特征值所用的对分法；还有些算法是用表格加算例表述的，例如，牛顿插值多项式系数计算，不少书中仅给出了一个具体算例的差商表．一般的计算方法书中，绝大多数算法虽然都是用数学公式表示的，但是未明确给出计算过程和计算条件，不能直接翻译成计算机语言．

将算法用带计算过程和计算条件的数学公式表示，不仅方便于程序设计，也便于手算．

2．纠正了一般计算方法中不当的提法和结论．

现行教材中有些提法和结论是有问题的，有些提法不准确，不能体现实质．例如，在解方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的直接法的稳定性分析中，一般教材的结论是条件数 $\mathrm{cond}_p(\boldsymbol{A})$愈大，稳定性愈差，未能体现算法．本书的结论是：直接法的稳定性与算法有关，与方程组系数矩阵本身的特性有关，还与右端扰动大小有关，但与系数矩阵的条件数没有直接关系．这样既弄清了高斯列主元消元法、全主元法为什么能求解高斯消元法中所不能求解的问题或者误差很大的问题，又弄清了为什么平方根法、改进平方根法要求方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 中 $\boldsymbol{A}$ 的本身性能必须较好才能使用的原因，这样提法才和算法有关．又如在求解 $f(x)=0$ 的牛顿法收敛性定理中，一般认为定理［ $f(a)f(b)<0$， $f'(x)\neq 0$， $f''(x)$ 不变号， $f(x_0)f''(x)>0$ ］理论性强，用处大，而定理在根的邻域内牛顿法收敛且是二阶收敛实用性差．我们的结论正好相反，原因是牛顿法并不麻烦，但要鉴别一个二阶导数不变号尚未见可行的数值算法．恰恰相反，后一定理的可操作性好，因为对任何求根区间，我们总可以将之划分成等距的 n 等份，只要 n 充分大，任何子区间或者有一个根，或者无根，只要有根，该子区间就是根的邻域，而且前一个定理只能在 $[a,b]$ 上有一个根才管用，而后者可求所有根（单根）．再如在讲逆幂法的功能时，不少书讲逆幂法的功能是求 $\boldsymbol{A}$ 按模最小特征值及对应特征向量，实际上应强调是求 $\boldsymbol{A}^{-1}$ 的按模最大特征值及特征向量为好．因为数学上只关心按模最大特征值．工程物理问题只关心固有频

率（基频），或许还关心次频、第三频，所对应特征值为最大特征值、次大特征值等．实际上逆幂法并不单独使用，它总是和对称矩阵三对角化（镜面反射变换）对分法求指定特征值、原点平移配合使用的，它所求的特征值是 $\boldsymbol{A}$ 经过原点平移后的最小特征值，而不是 $\boldsymbol{A}$ 的最小特征值，一般教材中过分强调原点平移后会降低按模最大特征值和按模次大特征值之比，实际上原点平移可能改变特征值序号，还可能增大二者的比，在数值计算中也只和对分法和逆幂法配合使用才有意义．

3．强调实用和应用．

一般教材都讲迭代法可解大、中型代数方程组，但不介绍压缩存储．实际上若不压缩存储，无论是在算法复杂性上和内存开销上，迭代法都不及直接法，更不及变带宽平方根法或改进平方根法（一般书不介绍），只是对于高阶线性代数方程组计算误差小一些．本书增加了压缩存储算法，压缩存储算法的计算公式与对应的非压缩存储算法的计算公式相比并不复杂，其程序复杂程度比非压缩存储的还要小．

4．对绝大多数（除压缩存储迭代法外）算法都给出了手算算例，其计算结论都用程序做了验证．

书中除了压缩存储赛得尔迭代法和变带宽平方根法外，都给出了手算算例，并在算例中给出了算法计算过程，对压缩存储赛得尔迭代法及程序设计难度较大的算法，例如高斯全主元消元法、镜面反射变换等才附了程序．

计算方法是数学的分支，学生不做习题很难掌握算法，不给出算例也不方便学生做题，不易提高学生的抽象归纳能力，计算方法虽和程序设计相关，但两者不相等，所以不必每个算法都附程序．程序中的算例也不能代替手算算例．冯康先生所著书中在计算每个算法时都附了程序，但冯康先生的计算方法实质上是用文字书写的软件包，不主要作为教学用．本书给出的所有算例都经过了计算机程序的验证，确保无误．

5．增加了一些新算法．

本书增加了幂级数型插值多项式及泰勒级数数值算法，在推导 Adams 线性多步法时，直接使用了幂级数插值多项式，给出了公式推导过程，而其他教材只是讲可使用牛顿插值多项式推导线性多项式，却未给出推导过程（原因在于太麻烦）．

此外，本书给出了一般牛顿积分代数精确度的证明，给出了高斯积分代数精确度的简洁证明，对牛顿插值多项式的计算也做了改进（仿秦九韶计算）……

本书由张世禄和何洪英编写，编写书稿时得到了数学与信息学院领导的大力支持，得到了我的学生陈豫眉、熊华、谭代伦的帮助，这里特以致谢！

张世禄　何洪英

2010 年 6 月于四川南充

目　录

第 1 章　误差和算法选择

误差是一习惯用语，误差不是失误引起的差错．

1.1　误差概念

误差与失误无关，误差对于原始数据而言是指用于计算时它（们）与真实值之间的差；对于计算结果而言是指它（们）与理论结果之间的差．误差和准确值没有直接关系，准确值是存在的，但是人类所有文献和资料中的值都不是准确值．

1.1.1　误差分类

若按误差来源分类，误差可分为如下四类．

1．模型误差

对于任何实际计算问题，计算之前至少要建立数学模型．对于工程物理问题，在建立数学模型之前还要建立物理模型．物理模型和数学模型的建立都是经过高度简化和抽象的结果．简化和抽象自然和真实问题有了差异，这种差异称为模型误差．

例如，“方桌”面积可用算式

$$S = a^2 \tag{1.1-1}$$

计算．

$S = a^2$ 就是数学模型．世界上没有绝对的平面，也没有四边相等且四个角都是直角的几何正方形，因此式（1.1-1）是简化和抽象的结果．这也是将方桌打上引号的原因．若一定按真实桌面计算面积，则需要做重积分运算，而边界条件是一条无法用解析算式描述的空间封闭曲线．显然，按式（1.1-1）所算出的面积值是可接受的且有意义的，真正的面积至少是目前算不出来的．

又如某研究站研制一种冶炼控制系统．研制人员将冶炼的材料视为绝对绝热的，遵守热力学第一定律．同时将炉体当做无限长的直圆柱．这样就把一个三维的边界复杂的热学问题变成了一个一维的可用热传导方程描述的简单问题：

$$\begin{cases} \dfrac{\partial u}{\partial t} = a^2 \dfrac{\partial^2 u}{\partial r^2} \\ \text{边界条件略} \end{cases} \tag{1.1-2}$$

世界上没有绝对绝热的材料，也没有几何中的直圆柱，更不可能无限长，但按式（1.1-2）

可及时算出炉体中半径为 r 处任何时刻的相关参数值，而且比人工从炉中用特制器皿取出溶液再由人工送到化验地化验后得到相关值既“及时”又“准确”得多．当然，按式（1.1-2）也不可能得到准确值．

计算方法里不考虑模型误差，或者说认为所做物理模型和数学模型都是合理的，或是满足用户要求的．

2．测试误差

所有实际计算问题在计算前都有一定量的已知数据，这些数据绝大多数都是由仪器仪表测量出来的．所有仪器仪表都有测量精度．测量精度决定于该仪器仪表的最小刻度或最小量度．

比最小刻度或最小量度小的部分只能用四舍五入法表示．由此带来的误差称为测试误差．测试误差为该仪器仪表的最小刻度或量度的一半．例如木匠用卷尺或角尺测量家具长度，卷尺和角尺的最小刻度为 1 mm，测试误差为 0.5 mm．测试误差实际上就是最大测试误差，常用绝对值表示．

3．舍入误差

所有工程物理问题的计算都不是人工完成的，而是计算机完成的．所有数都存放在计算机的存储器里（对于较大型问题及数据较多的问题，原始数据先以文件方式存入外存储器，程序运行时再读入内存器），存储器由若干字节组成，一个数可存放在一个字节里，也可存放在两个字节里，还可存放在 4 个字节里，甚至可存放在 16 个字节里（4 倍精度数）．即使存放在 16 个字节里，也不可能存放无理数或超过 16 个字节所允许的长度的更长数．在计算机中，整数或长整数是没有误差的．但实（型）数是用尾数和指数表示的．尾数中第 1 个数表示数符，指数中第 1 个数为阶符，尾数和指数的数字（二进制）个数是有限的．比尾数所容许的个数多的数后面的数字只能舍或入，若尾数（去掉数符）共 k 位，则对有 $k+1$ 位数字（后面介绍）的数的第 $k+1$ 位用舍入处理（转换成十进制数后为四舍五入）．对于阶比指数中所允许的最大值还大的数，当阶符为正时，溢出．当阶符为负时，以 0 表示．由此所产生的误差称为舍入误差．

4．截断误差

计算方法由两部分组成．其一是数值代数，其二是数值逼近．对于数值逼近部分，常用多项式或有理式代替原来的函数，或者用多项式代替原来的函数再计算，截断误差是算法本身的误差．

高等数学中有计算公式

$$e=\sum_{i=0}^{\infty}\frac{1}{i!}=\sum_{i=0}^{n}\frac{1}{i!}+\sum_{i=n+1}^{\infty}\frac{1}{i!}.$$

实际计算时，只能用

$$e \approx \sum_{i=0}^{n} \frac{1}{i!}.$$

$\sum_{i=n+1}^{\infty} \frac{1}{i!}$ 就是截断误差.

1.1.2　误差表示法和误差限

1．误差表示法

误差大小可用绝对误差和相对误差两种形式表示.

【定义 1】若 x^* 为理论值或真实值，x 为计算值或测量值，则

$$\varepsilon(x) = \Delta x = x^* - x \tag{1.1-3}$$

称为绝对误差.

人们使用绝对误差时，通常用其绝对值. 只有少数领域要考虑符号. 例如在机械设计时，对于活塞和汽缸的加工，若以汽缸为准，则活塞的直径只能比汽缸直径小，误差（公差）只能为负.

单有绝对误差还不够，还须定义相对误差.

【定义 2】若 x^* 为理论值或真实值，x 为计算值或测量值，则

$$\varepsilon_r(x) = \frac{\varepsilon(x)}{x^*} = \frac{\Delta x}{x^*} \tag{1.1-4}$$

为相对误差. 由于 x^* 是不知道的，所以式（1.1-4）常表述成 $\frac{\Delta x}{x}$，相对误差的具体值也是不知道的，也是算不出来的，只能给出一个范围或量级.

2．误差限

误差限也有绝对误差限和相对误差限之分.

所谓绝对误差限，是指计算某一问题时，用户所允许的（绝对值）最大绝对误差. 绝对误差限可用 η 表示：

$$\eta = \max |\varepsilon(x)| = \max |x^* - x| \tag{1.1-5}$$

所谓相对误差限，是指计算某一问题时，用户所允许的（绝对值）最大相对误差. 相对误差限可用 η_r 表示：

$$\eta_r = \max |\varepsilon_r(x)| \tag{1.1-6}$$

数学模型确定后，选择算法的第一依据就是误差限. 软件人员选择算法时，该算法若有截断误差，那么截断误差不能超过误差限. 对于用仪器仪表所测得的各数据，不仅开始时不允许超过用户所给定的误差限，同时由测试误差而引起的计算误差也不能超过误差限. 为此须选择足够精度的仪器仪表，程序中也应根据误差限确定是否使用双精度数或 4 倍精度数.

前面介绍的四类误差通常称为先天误差，而算法和计算过程因原有初始误差而派生出的计算误差属于后天误差. 后天误差的大小由算法稳定性决定.

1.1.3 误差运算

由于绝对误差和相对误差的具体值都是不知道的，故误差运算只能是近似计算．可以将 Δx 作为 $\mathrm{d}x$ 处理．计算方法里没有专门介绍误差运算的算法．但是在算法选择中要用到误差运算知识．

将 $\varepsilon_r(x)$ 近似表示成

$$\varepsilon_r(x)=\frac{\Delta x}{x^*}\approx\frac{\mathrm{d}x}{x}=\mathrm{d}\ln x \tag{1.1-7}$$

当 $x<0$ 时，$\dfrac{\mathrm{d}x}{x}=\dfrac{\mathrm{d}x'}{x'}=\mathrm{d}\ln x'$（$x'=-x$），将 x' 换名为 x，则又变成了式（1.1-7）．为了叙述方便，计算时式（1.1-7）中“$\approx$”用“=”代替．书中大多数“=”实际上都是“$\approx$”号，这点以后不再提及．

1．乘积的相对误差

乘积的相对误差为

$$\varepsilon_r(x_1x_2)=\mathrm{d}\ln x_1x_2=\mathrm{d}(\ln x_1+\ln x_2)=\varepsilon_r(x_1)+\varepsilon_r(x_2) \tag{1.1-8}$$

2．商的相对误差

由式（1.1-8）可直接得出

$$\varepsilon_r\left(\frac{x_1}{x_2}\right)=\varepsilon_r(x_1)-\varepsilon_r(x_2) \tag{1.1-9}$$

3．和的相对误差

和的相对误差为

$$\begin{aligned}\varepsilon_r(x_1+x_2)&=\frac{\mathrm{d}(x_1+x_2)}{x_1+x_2}=\frac{\mathrm{d}x_1}{x_1+x_2}+\frac{\mathrm{d}x_2}{x_1+x_2}\\&=\frac{x_1}{x_1+x_2}\frac{\mathrm{d}x_1}{x_1}+\frac{x_2}{x_1+x_2}\frac{\mathrm{d}x_2}{x_2}\\&=\frac{x_1}{x_1+x_2}\varepsilon_r(x_1)+\frac{x_2}{x_1+x_2}\varepsilon_r(x_2)\end{aligned} \tag{1.1-10}$$

由式（1.1-10）可以看出，当 $x_1\approx -x_2$ 时，$\varepsilon_r(x_1+x_2)$ 会相当大．因此要避免两个相近数相减．另外，当 $x_1\gg x_2$ 时，$\varepsilon_r(x_1+x_2)\approx\varepsilon_r(x_1)$．

1.1.4 有效数字

在计算机中数字只有整型数（含长整型与短整型）和实型数两种，整型数都是整数，整数中的所有数字都是有效数字．有效数字只针对实型数．计算机算出的数和输出的数的数字不一定都是有效数字．有效数字通常与测试误差和误差限有关．

【定义3】近似值 x 准确到小数点后第 n 位，或者称从小数点后第 n 位数字到 x 最左边的

第一个非零数字的全部为有效数字.

由定义有$\left|x-x^*\right| \leqslant \frac{1}{2}\times 10^{-n}$.

【例 1】已知$\mathrm{e}=2.71828\cdots$，问下列 e 的近似值有几位有效数字?

（1）2.7　　（2）2.7183

【解】（1）$\left|\mathrm{e}-2.7\right|=0.01828\cdots<\frac{1}{2}\times 10^{-1}$.

（2）$\left|\mathrm{e}-2.7183\right|=0.00002\cdots<\frac{1}{2}\times 10^{-4}$.

由计算结果知 2.7 准确到小数点后 1 位，而 2.7183 准确到小数点后第 4 位.

【例 2】指出下列近似值的有效数字并估计最大绝对误差.

1.02　　1.020　　381.70　　0.035

【解】由定义有 1.02, 1.020, 381.70, 0.035 的有效数字分别为 3 位、4 位、5 位和 2 位.

这里指出:

（1）在过程控制系统中使用传感器、数模转换器、模数转换器等外设时，其测试精度应和用户要求的有效数字位数一致.

（2）科技计算中要求保留到小数点后 6 位以上数字的题目不多，用单精度数足够，对于高精度计算问题则可将所涉及变量定义成双倍精度数或 4 倍精度数. 程序中不必考虑有效数字位，因为一切计算是由计算机担任的. 考虑有效数字位数只会增加运算时间和程序编写难度，只须在输出时做处理. 按用户给出的有效位数输出则可. 书中相当多的结果，除了考虑篇幅外，大多数都不考虑有效数字位数.

（3）在计算机中，实（型）数是用浮点数存放的，若将之翻译成十进制数，则上面 4 个数应为0.102×10^{1}，0.1020×10^{1}，0.38170×10^{3}和0.35×10^{-1}. 这正是 0.035 的有效数字位为 2 而不为 3 的原因.

注意：手算时有效数字取得愈多，计算量愈大，出错概率愈大. 手算时所取数的位数最好和给定精度（误差限）一致，不必像计算机那样取位太多.

1.2　算法选择

对于简单数学问题，数学模型就是计算公式，但对于工程物理问题，数学模型确定后还需要选择计算方法，算法选择原则如下.

1.2.1　正确性

计算方法中的正确性不是理论上的正确性，而是结果的正确性. 结果正确包括如下几个方面.

1．误差不超过误差限

不少算法有截断误差，截断误差仅是误差的一部分．显然，截断误差若超过误差限，则算出的结果无意义．

对于工程物理问题，由于物理模型和数学模型是经过高度简化和抽象后的产物，所以只存在理论解而不存在准确解．由于舍入误差和测试误差的存在，即便算法没有截断误差，也找不出理论解．

计算方法中只有两类算法，一类是迭代法，另一类是递推算法．对于迭代法，误差不超过误差限是通过算式

$$\begin{cases} |x_{n+1}-x_n|>\varepsilon \\ \left\|\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^{(n)}\right\|>\varepsilon \end{cases},\quad n=1,2,3,\cdots \tag{1.2-1}$$

决定的．

式（1.2-1）中 x_{n+1} 表示第 $n+1$ 次迭代所得的解，x_n 是第 n 次迭代所得的解．ε 为迭代精度，即绝对误差限．题目中要求的有效数字位数和误差限意义一样．而 $\boldsymbol{x}^{(n+1)}$ 是第 $n+1$ 次迭代所得到的解向量，$\boldsymbol{x}^{(n)}$ 是第 n 次迭代所得到的解向量．“$\|\ \|$”表示范数（以后介绍）．一旦 $|x_{n+1}-x_n|<\varepsilon$ 或 $\left\|\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^{(n)}\right\|<\varepsilon$，则表示误差在误差限内计算结束，测试误差也不能超过误差限．

一般科技计算问题中，原始数据（输入数据）由用户提供，当原始数据（有效数字）的精度低于用户要求的误差限时，计算结果无意义，须要求用户用更高精度的测量工具重新测量. 对于实时处理系统，如过程控制系统，系统设计者必须购置能满足误差限的测量仪器（传感器、数模转换器、模数转换器等），否则计算结果将无意义．

2．收敛于理论解

计算方法中有一类算法是迭代法，对于迭代算法，用于某个问题会收敛，但用于另一问题则可能会发散．所谓收敛，是在式（1.2-1）中找得到一个 n，使得 $|x_{n+1}-x_n|<\varepsilon$ 或 $\left\|\boldsymbol{x}^{(n+1)}-\boldsymbol{x}^{(n)}\right\|<\varepsilon$．否则称为发散．只有收敛的算法才有意义，在数学模型正确的前提下，收敛的算法自然收敛于理论解．

3．及时得出解

计算方法不追求理论解，只追求满意解．满意解不仅应是在用户给定的误差限内的解，而且还应是在时间限内的解．式（1.2-2）是一维偏微分方程，用该微分方程的原因有两条：

（1）误差在误差限内；

（2）能及时得到解．

有些问题对“及时”要求很高，例如歼击机上都安装有大气数值计算机，大气数值计算机实际上是一片具有专门功能的单片机．它配备有专门程序，该程序必须及时算出敌我双方

飞机的位置、速度、温度和气压，及时决定攻击或防卫．对于这类问题，及时最为重要．

4．算法稳定

稳定性是计算方法中的专用名词．

【定义 1】若使用某算法在计算之前初始误差为 ε，整个计算过程误差都小于 $k\varepsilon$，k 是大于 0 的某个常数，则称算法是稳定的．

这一定义和具体问题无关．

【定义 2】用户给定的误差限为 η．若算法的初始误差为 ε，只要计算过程中计算误差不超过 η'，且 $\eta' \leqslant \eta$，则算法是稳定的．或者说当计算误差不超过截断误差时算法是稳定的．截断误差不允许超过误差限．

不稳定的算法是无法得到正确解的．理论上能得到理论解的算法不一定是好算法．

【例 1】计算 $S_n = \int_0^1 \frac{x^n}{5+x}\mathrm{d}x$．

【解】对于任意 n 难以直接用牛顿-莱布尼兹公式算出积分值．不过由本例可通过下述过程算出：

$$S_0 = \int_0^1 \frac{x^0}{5+x}\mathrm{d}x = \ln(5+x)\Big|_0^1 = \ln\frac{6}{5}.$$

另外，我们还可得到

$$5S_{n-1} + S_n = 5\int_0^1 \frac{5x^{n-1}}{5+x}\mathrm{d}x + \int_0^1 \frac{x^n}{5+x}\mathrm{d}x = \int_0^1 x^{n-1}\mathrm{d}x = \frac{1}{n}.$$

由此可得

$$\begin{cases} S_0 = \ln\dfrac{6}{5} \\ S_n = \dfrac{1}{n} - 5S_{n-1} \end{cases} \tag{1.2-2}$$

表 1.1 列出了用式（1.2-2）计算出的 $S_1 \sim S_{20}$．

表 1.1　用式（1.2-2）计算出的 $S_1 \sim S_{20}$

i	S_i	i	S_i	i	S_i	i	S_i
1	0.088392	6	0.024428	11	−0.307181	16	1003.922729
2	0.058039	7	0.020719	12	1.619237	17	−5019.554688
3	0.043138	8	0.021407	13	−8.019263	18	25097.828125
4	0.034310	9	0.004076	14	40.167744	19	−125489.085938
5	0.028448	10	0.079618	15	−200.772049	20	627445.500000

从式（1.2-2）中可看出 $1 > S_n > 0$，$S_n < S_{n-1}$．但是表中的值和理论分析差异甚大，从 S_8 开始以后的数据和理论分析已不一致，S_{11} 已小于 0，S_{12} 已大于 1，愈到后面愈不好理解．

实际上，我们所取 $S_0 = 0.182322 \neq \ln\frac{6}{5}$．在计算机中对数函数是用迭代法计算的，只能

得出近似值．设所得值与理论值之差为 ε，通过式（1.2-2），计算 S_1 时误差为 -5ε．若不考虑其他计算误差，则 S_n 的计算误差为 $(-1)^n 5^n \varepsilon$．当 n 较大时，$(-1)^n 5^n \varepsilon$ 的绝对值会非常大，远远超过 1，且数符交错．

由式（1.2-2），当 $n>1$ 时有 $5S_{n-1} < S_n + 5S_{n-1} < 6S_{n-1}$．考虑到 $S_n + 5S_{n-1} = \dfrac{1}{n}$，当 $n=21$ 时有

$$\frac{1}{6\times 21} < S_{20} < \frac{1}{5\times 21}.$$

取 $S_{20} = \dfrac{1}{2}\left(\dfrac{1}{6\times 21} + \dfrac{1}{5\times 21}\right) \approx 0.008730$，将式（1.2-2）改写成

$$\begin{cases} S_{n-1} = -\dfrac{S_n}{5} + \dfrac{1}{5n} \\ S_{20} = \dfrac{1}{2}\left(\dfrac{1}{6\times 21} + \dfrac{1}{5\times 21}\right) \approx 0.008730 \end{cases} \tag{1.2-3}$$

按式（1.2-3）计算的结果见表 1.2.

表 1.2　用式（1.2-3）计算的结果

i	S_i	i	S_i	i	S_i	i	S_i
1	0.088392	6	0.024325	11	0.014071	16	0.009898
2	0.058039	7	0.021233	12	0.012977	17	0.009336
3	0.043139	8	0.018837	13	0.012040	18	0.008876
4	0.034306	9	0.016926	14	0.011229	19	0.008254
5	0.028468	10	0.015368	15	0.010520	20	0.008730

由式（1.2-3）可看出，若 S_{20} 和理论值之差为 ε，则 S_n 的误差为 $(-1)^{20-n}(\varepsilon/5)^{20-n}$，$n=19,18,\cdots,1$．按定义 1，式（1.2-3）是不稳定的算法，而按定义 2 是稳定算法．

因现在尚未正式介绍计算方法，故无法利用定义 2 判别算法的稳定性．

上面两表中的计算结果都是由单精度数算出的．

1.2.2　选择低复杂性算法

计算方法所介绍的算法都是解决一类问题的通用算法，算法的复杂性是指算法中主要计算的计算量．主要计算是指算法中最耗费机时运算的量级．在程序中一次函数运算（绝对值计算例外）比乘除运算所耗机时量级高，一次乘除运算比简单运算（简单运算是指加减运算、大小比较运算和数据交换运算）量级高．通常一个算法会包含三类运算，但只考虑主要运算．当三类运算量级都一样时，只考虑函数运算．当乘除运算比函数运算次数的量级高时，只考虑乘除运算，当主要运算是简单运算时，则只考虑简单运算的量级．

算法复杂性通常用 $O(N^k)$ 或 aN^k 表示．也有用 $O(N^k \ln N)$ 表示的．

变高复杂性算法为低复杂性算法有两种途径．其一是简化数学模型，使用满足用户给出

的误差限的低复杂性算法（前面已有例子介绍）．其二是由计算方法的学者给出低复杂性算法，例如快速傅里叶变换．

算法复杂性有时还指内存开销．为了降低内存的开销，可采取的办法有

（1）压缩存储；

（2）分块存储，分块运算．

通常，计算方法教材不介绍降低内存开销的方法．本书从实用性出发介绍压缩存储．

注意：当前计算机内存都相当大，即使是微机，内存为 1 GB 的也不鲜见．但在考虑内存开销时，却不能只考虑内存大小，还要考虑所用计算机语言的寻址方式．由于这些涉及计算机专业知识，这里不予介绍．

1.2.3　减少误差的一些简单办法

有些算式可利用恒等变换来避免两个相近数相减．

【例 2】 求二次方程 $x^2+px+q=0$ 的解，其中若 $p^2 \gg q$，$p>0$．传统计算公式为

$$\begin{cases} x_1=\dfrac{1}{2}(-p+\sqrt{p^2-4q}) \\ x_2=\dfrac{1}{2}(-p-\sqrt{p^2-4q}) \end{cases} \tag{1.2-4}$$

式（1.2-4）中 x_1 的算式出现了两个相近数相减，但若用恒等变换使之变成

$$x_1=\frac{-2q}{p+\sqrt{p^2-4q}} \tag{1.2-5}$$

就不会出现两个近似数相减了．

【例 3】 设 $A=(10^5+a)^3$，$B=(10^5+b)^3$，$a\neq b$，且均是一位数字，计算 $A-B$．

【解】 A 和 B 值接近，也属两个相近数相减．若直接按算式先算 A 后算 B，再算 $A-B$，既会产生较大的相对误差，又会丢失有效数字．利用恒等变换

$$\begin{aligned}(10^5+a)^3-(10^5+b)^3&=(10^5+a-10^5-b)((10^5+a)^2+(10^5+a)(10^5+b)^2+(10^5+b)^2)\\&=(a-b)((10^5+a)^2+(10^5+a)(10^5+b)^2+(10^5+b)^2)\end{aligned}$$

右端的算式计算误差小，且不易丢失有效数字．

这类算式还有一些，所用技巧都属于中学数学范畴，这里不一一列举．

1.2.4　一种新的算法模式

这里所说的算法新模式是指给出带计算过程、计算条件、计算范围并且与程序设计语言有一一对应关系的算法．

计算方法不是人手算的算法，而是计算机使用的算法．计算机只能识别人们编写的程序，程序也是算法，程序是计算机使用的算法，程序设计就是将非计算机语言描述的算法翻译成计算机语言表述的算法，带计算过程、计算条件、计算范围的算法容易翻译成程序．

【例 4】秦九韶法计算多项式的计算公式为

$$P_n(x)=(\cdots(a_nx+a_{n-1})x+\cdots+a_1)x+a_0 \tag{1.2-6}$$

这一公式不易编出程序，但若将之改写成

$$\begin{cases} S_n=a_n \\ S_i=S_{i+1}x+a_i, \quad i=n-1,n-2,\cdots,0 \end{cases} \tag{1.2-7}$$

则很容易编出程序.

【例 5】Newton 插值多项式的计算公式为

$$N_n(x)=c_0+c_1(x-x_0)+\cdots+c_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}) \tag{1.2-8}$$

若将之改写成

$$\begin{cases} S_n=c_n \\ S_i=S_n(x-x_i)+c_{i-1}, \quad i=n-1,n-2,\cdots,1 \end{cases} \tag{1.2-9}$$

这一算法也易编写出程序，程序的复杂程度和算法的复杂性都比直接用式（1.2-8）所编写的程序的复杂度和算法的复杂性低.

在一般计算方法书中还有用图表表述的算法，一些应用数学书中还有用自然语言表述的算法，本书所有算法都用带计算过程、计算条件和计算范围的数学公式表述，算式和程序设计语言有一对一的关系.

算法表述通常和计算误差关系不大，但是若改变了计算顺序和计算过程，则计算误差会改变. 本小节的两例算式——式（1.2-7）和式（1.2-9），都比原多项式计算和 Newton 插值多项式计算的算法稳定性好.

习题 1

1. 设 $x_1=1.23$，$x_2=3.64$，$x_3=9.85$ 均准确到末位数字，试估计由这些数据计算 $x_1x_2+x_3$ 的相对误差.
2. 设 $x>0$，x 的相对误差限为 δ，分别求 x^n 和 $\ln x$ 的相对误差限.
3. 正方形的边长约为 100 cm，测量时误差最大达到多少时，才能使面积的误差不超过 1 cm^2？
4. 下列各数均是由四舍五入得到的近似值，试指出各数具有几位有效数字及它们的绝对误差限和相对误差限.

   （1）0.024，　（2）0.4135，　（3）57.5，　（4）6000，　（5）−2000.00.
5. 取 $x=12.49$，x 的近似值 12.5, 12.4 和 12.48 分别有几位有效数字？
6. 如何计算下列各值，使其误差较小？

   （1）当 $|x|\ll 1$ 时，$f(x)=\dfrac{1-\cos x}{\sin x}$；（2）当 $f(x)=\int_N^{N+1}\dfrac{\mathrm{d}x}{1+x^2}$ 时（其中 N 充分大）.
7. 计算积分 $y_n=\int_0^1\dfrac{x^n}{x+100}\mathrm{d}x$，$n=0,1,2,3,4$ 的近似值.

第 2 章　解线性方程组方法之直接法

解线性方程组的数值算法有两类，其一是直接法，其二是迭代法，本章介绍直接法．

直接法多用于求解中、小型线性代数方程组，也就是说 500 阶以下的方程组．直接法又可以分为三个子类：Gauss 消元法、三角分解法和正交变换法．考虑到课时的关系，本书不介绍正交变换法，也不介绍 Gauss-Jordan 消元法．

2.1　Gauss 消元法

消元法包括 Gauss 消元法、Gauss 列主元消元法和全主元素消元法，以及 Gauss-Jordan 消元法，本书只介绍前三种消元法．三个 Gauss 消元法的计算过程都由消元和回代组成，消元都是利用矩阵的初等变换，即方程组的同解变换，目的是使一般矩阵变成上三角矩阵便于用代入消元法求解，本教材将之变成单位上三角矩阵．回代仍是利用中学数学知识——代入消元．

2.1.1　Gauss 消元法

Gauss 消元法的计算过程由两部分组成．

1．消元

消元所用知识从代数角度上看，是对系数矩阵的初等变换，说得更明白一些就是只用了中学的数学知识．

（1）将方程两端同除以一个非零常数，解不变；

（2）将一个方程乘以一个常数和另一个方程相加，解不变．

对于方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，$\boldsymbol{A}=\{a_{ij}\}$，令

$$a_{ij}^{(0)}=a_{ij},\ \ i,j=1,2,\cdots,n.$$

$$b_i^{(0)}=b_i,\ \ i=1,2,\cdots,n.$$

即

$$\begin{cases} a_{11}^{(0)}x_1+a_{12}^{(0)}x_2+\cdots+a_{1k}^{(0)}x_k+\cdots+a_{1n}^{(0)}x_n=b_1^{(0)} \\ a_{21}^{(0)}x_1+a_{22}^{(0)}x_2+\cdots+a_{2k}^{(0)}x_k+\cdots+a_{2n}^{(0)}x_n=b_2^{(0)} \\ \vdots \\ a_{j1}^{(0)}x_1+a_{j2}^{(0)}x_2+\cdots+a_{jk}^{(0)}x_k+\cdots+a_{jn}^{(0)}x_n=b_j^{(0)} \\ \vdots \\ a_{n1}^{(0)}x_1+a_{n2}^{(0)}x_2+\cdots+a_{nk}^{(0)}x_k+\cdots+a_{nn}^{(0)}x_n=b_n^{(0)} \end{cases} \tag{2.1-1}$$

消元过程如下：

① 将第一个方程两端同除以 $a_{11}^{(0)}$ 得

$$\begin{cases} x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1k}^{(1)}x_k + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ a_{1k}^{(1)} = a_{1k}^{(0)} \big/ a_{11}^{(0)} \\ b_1^{(1)} = b_1^{(0)} \big/ a_{11}^{(0)} \end{cases} \qquad k = 2,3,\cdots,n\ .$$

② 将当前的第一个方程两端同乘以 $-a_{j1}^{(0)}$ 并与第 j 个方程相加得

$$\begin{cases} 0 \cdot x_1 + a_{j2}^{(1)}x_2 + \cdots + a_{jk}^{(1)}x_k + \cdots + a_{jn}^{(1)}x_n = b_j^{(1)} \\ a_{jk}^{(1)} = a_{jk}^{(0)} - a_{1k}^{(1)}a_{j1}^{(0)} \\ b_j^{(1)} = b_j^{(0)} - b_1^{(1)}a_{j1}^{(0)} \end{cases} \qquad \begin{matrix} j = 2,3,\cdots,n \\ k = 2,3,\cdots,n \end{matrix}$$

此时方程组变成

$$\begin{cases} x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1k}^{(1)}x_k + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ \qquad a_{22}^{(1)}x_2 + \cdots + a_{2k}^{(1)}x_k + \cdots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad\qquad \vdots \\ \qquad a_{n2}^{(1)}x_2 + \cdots + a_{nk}^{(1)}x_k + \cdots + a_{nn}^{(1)}x_n = b_n^{(1)} \end{cases} \tag{2.1-2}$$

式（2.1-2）表示完成了第一轮消元.

现假设已进行了 $i-1$ 轮消元，方程组已变成

$$\begin{cases} x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1i}^{(1)}x_i + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)} \\ \qquad x_2 + \cdots + a_{2i}^{(2)}x_i + \cdots + a_{2n}^{(2)}x_n = b_2^{(2)} \\ \qquad\qquad \vdots \\ \qquad\qquad a_{ii}^{(i-1)}x_i + \cdots + a_{in}^{(i-1)}x_n = b_i^{(i-1)} \\ \qquad\qquad a_{i+1i}^{(i-1)}x_i + \cdots + a_{i+1n}^{(i-1)}x_n = b_{i+1}^{(i-1)} \\ \qquad\qquad \vdots \\ \qquad\qquad a_{ni}^{(i-1)}x_i + \cdots + a_{nn}^{(i-1)}x_n = b_n^{(i-1)} \end{cases}$$

第 i 轮消元计算过程如下：

① 将第 i 个方程两端同除以 $a_{ii}^{(i-1)}$ 得

$$\begin{cases} x_i + a_{ii+1}^{(i)}x_{i+1} + \cdots + a_{in}^{(i)}x_n = b_i^{(i)} \\ a_{ik}^{(i)} = a_{ik}^{(i-1)} \big/ a_{ii}^{(i-1)} \\ b_i^{(i)} = b_i^{(i-1)} \big/ a_{ii}^{(i-1)} \end{cases} \qquad k = i+1, i+2, \cdots, n \tag{2.1-3}$$

② 将当前的第 i 个方程乘以 $-a_{ji}^{(i-1)}$ 并与第 j 个方程相加得

$$\begin{cases} a_{ji+1}^{(i)}x_{i+1}+a_{ji+2}^{(i)}x_{i+2}+\cdots+a_{jn}^{(i)}x_n=b_j^{(i)} & i=1,2,\cdots,n-1 \\ a_{jk}^{(i)}=a_{jk}^{(i-1)}-a_{ik}^{(i)}a_{ji}^{(i-1)} & j=i+1,i+2,\cdots,n \\ b_j^{(i)}=b_j^{(i-1)}-b_i^{(i)}a_{ji}^{(i-1)} & k=i+1,i+2,\cdots,n \end{cases} \tag{2.1-4}$$

对于式（2.1-3）和式（2.1-4），当$i=1,2,\cdots,n-1$时实际上是全程消元公式，经过$n-1$轮消元之后，方程组就变成了

$$\begin{cases} x_1+a_{12}^{(1)}x_2+\cdots+a_{1i}^{(1)}x_i+\cdots+a_{1n}^{(1)}x_n=b_1^{(1)} \\ \qquad x_2+\cdots+a_{2i}^{(2)}x_i+\cdots+a_{2n}^{(2)}x_n=b_2^{(2)} \\ \qquad\quad \ddots \\ \qquad\qquad x_i+\cdots+a_{in}^{(i)}x_n=b_i^{(i)} \\ \qquad\qquad\quad \ddots \\ \qquad\qquad\qquad a_{nn}^{(n-1)}x_n=b_n^{(n-1)} \end{cases}$$

由上式可得到

$$x_n=b_n^{(n-1)}\big/a_{nn}^{(n-1)} \tag{2.1-5}$$

当然也可认为

$$\begin{cases} a_{nn}^{(n)}=a_{nn}^{(n-1)}\big/a_{nn}^{(n-1)}=1 \\ b_n^{(n)}=b_n^{(n-1)}\big/a_{nn}^{(n-1)} \\ x_n=b_n^{(n)} \end{cases} \tag{2.1-6}$$

2．回代

计算出x_n后，由式（2.1-4）通过回代可逐步求出x_i，回代公式为

$$x_i=b_i^{(i)}-\sum_{j=i+1}^{n}a_{ij}^{(i)}x_j,\quad i=n-1,n-2,\cdots,1 \tag{2.1-7}$$

2.1.2　Gauss 消元法的计算过程和计算算例

第一步：输入系数矩阵a_{ij}，输入右端项b_i，$i=1,2,\cdots,n$，$j=1,2,\cdots,n$.

第二步：用式（2.1-4）消元，用式（2.1-5）或式（2.1-6）求解x_n，用式（2.1-7）回代求解.

【例 1】用 Gauss 消元法求解方程组

$$\begin{cases} 2x_1+4x_2+6x_3=-2 \\ x_1+3x_2+x_3=-4 \\ 2x_1+x_2+2x_3=-3 \end{cases}$$

【解】按 Gauss 消元法计算公式（2.1-3）和公式（2.1-4），消元过程为

将第一个方程两端同除以 2 得

$$x_1+2x_2+3x_3=-1\,.$$

将新的第一个方程乘以 -1 并与第二个方程相加得新的第二个方程

$$x_2-2x_3=-3\,.$$

将新的第一个方程乘以 -2 并与第三个方程相加得新的第三个方程

$$-3x_2-4x_3=-1\,.$$

将新的第二个方程乘以 3 并与新的第三个方程相加得新的第三个方程

$$-10x_3=-10\,.$$

按式（2.1-5）得

$$x_3=1\,.$$

代入式（2.1-7）得

$$x_2=-1\,.$$

$$x_1=-2\,.$$

按上面的格式手算显得很烦琐，不如借用增广矩阵给出各轮消元结果.

【例 2】用 Gauss 消元法求解方程组

$$\begin{cases}2x_1+x_2+2x_3+x_4=14\\ x_1+2x_2-2x_3+x_4=3\\ 3x_1+x_2+x_3+2x_4=16\\ 2x_1+x_2+x_3+x_4=11\end{cases}$$

【解】第一轮消元结果是

$$\begin{bmatrix}1&0.5&1&0.5&7\\1&2&-2&1&3\\3&1&1&2&16\\2&1&1&1&11\end{bmatrix}\rightarrow\begin{bmatrix}1&0.5&1&0.5&7\\0&1.5&-3&0.5&-4\\0&-0.5&-2&0.5&-5\\0&0&-1&0&-3\end{bmatrix}$$

第二轮消元结果是

$$\begin{bmatrix}1&0.5&1&0.5&7\\0&1&-2&1/3&-8/3\\0&-0.5&-2&0.5&-5\\0&0&-1&0&-3\end{bmatrix}\rightarrow\begin{bmatrix}1&0.5&1&0.5&7\\0&1&-2&1/3&-8/3\\0&0&-3&2/3&-19/3\\0&0&-1&0&-3\end{bmatrix}$$

第三轮消元结果是

$$\begin{bmatrix}1&0.5&1&0.5&7\\0&1&-2&1/3&-8/3\\0&0&-3&2/3&-19/3\\0&0&-1&0&-3\end{bmatrix}\rightarrow\begin{bmatrix}1&0.5&1&0.5&7\\0&1&-2&1/3&-8/3\\0&0&1&-2/9&19/9\\0&0&0&-2/9&-8/9\end{bmatrix}$$

利用回代公式得

$$\begin{cases} x_4 = 4 \\ x_3 = \dfrac{19}{9} + \dfrac{8}{9} = 3 \\ x_2 = -\dfrac{8}{3} - \dfrac{4}{3} + 2 \times 3 = 2 \\ x_1 = 7 - 0.5 \times 4 - 1 \times 3 - 0.5 \times 2 = 1 \end{cases}$$

Gauss 消元法不是一个实用算法，不少很容易求出解的方程组用该算法不能得到解.

【例 3】用 Gauss 消元法求解方程组

$$\begin{cases} x_2 = 1 \\ x_1 + 2x_2 = 3 \end{cases}$$

【解】因 $a_{11}^{(0)} = a_{11} = 0$，不能用 Gauss 消元法求解.

【例 4】用 Gauss 消元法求解方程组

$$\begin{cases} 10^{-20} x_1 + x_2 = 1 + 10^{-20} \\ x_1 + x_2 = 2 \end{cases}$$

【解】在计算机里，若用单精度数甚至用双精度数存放上式，都只能得到下面的方程组：

$$\begin{cases} 10^{-20} x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$$

用 Gauss 消元法求解得

$$\begin{bmatrix} 1 & 10^{20} & 10^{20} \\ 1 & 1 & 2 \end{bmatrix} \to \begin{bmatrix} 1 & 10^{20} & 10^{20} \\ 0 & -10^{20} & -10^{-20} \end{bmatrix}$$

解之得

$$\begin{cases} x_2 = 1 \\ x_1 = 0 \end{cases}$$

显然，这不是原方程组的满意解.

2.1.3　Gauss 消元法计算量

Gauss 消元法由消元和回代两部分组成，计算量也由两部分组成，由于 Gauss 消元法中加减法的计算量级和乘除法的计算量级相同，故只考虑乘除法的计算量.

1．消元计算量

由式（2.1-3）知 Gauss 消元法要用到的除法次数为

$$Q_{11} = n + (n-1) + \cdots + 1 = \frac{1}{2}(n^2 + n) .$$

由式（2.1-4）得所用的乘法次数为

$$Q_{12}=\sum_{i=1}^{n-1}((n-i)^2+(n-i))$$
$$\doteq\frac{1}{6}n(n-1)(2n-1)+\frac{1}{2}n(n-1)$$
$$\doteq\frac{1}{3}n^3$$

由此知 Gauss 消元法消元的计算量为 $\frac{1}{3}n^3$.

2．回代计算量

由式（2.1-5）和式（2.1-6）知回代计算量为

$$Q_2\doteq\sum_{i=1}^{n-1}\frac{n^2}{2}.$$

由上面的分析知 Gauss 消元法的总计算量为 $\frac{1}{3}n^3$ 次乘法.

本书所用计算公式和一般教材所用公式略有不同，其计算量量级和一般教材一样（计算量略低），给出该算法的目的在于便于编写程序，便于做稳定性分析.

2.1.4 Gauss 列主元素消元法

【定义 1】 $\left|a_{pi}\right|=\max\limits_{j}\left|a_{ji}\right|$，$i=1,2,\cdots,n-1$，$j=i,i+1,\cdots,n$，称为第 i 列的列主元素.

Gauss 列主元素消元法基于 Gauss 消元法，它是在第 i 轮消元时增加选列主元素并将列主元素所在行的元素与第 i 行元素交换的一种计算．由于交换两方程的位置（序号）解不变，因而在无舍入误差和测试误差的前提下，Gauss 列主元素消元法所求解仍是理论解.

Gauss 列主元素消元法计算公式和计算过程如下.

1．第 i 轮消元（$i=1,2,\cdots,n-1$）计算步骤和计算公式

（1）找第 i 列主元素行的行号 p：

$$\begin{cases}p=i\\ p=j,\quad \left|a_{ji}^{(i-1)}\right|>\left|a_{pi}^{(i-1)}\right|\end{cases}\qquad j=i+1,i+2,\cdots,n \tag{2.1-8}$$

（2）第 i 行和第 p 行交换（$p=i$ 时不交换）：

$$\begin{cases}a_{pk}^{(i-1)}\Leftrightarrow a_{ik}^{(i-1)}\\ b_{p}^{(i-1)}\Leftrightarrow b_{i}^{(i-1)}\end{cases}\qquad k=i,i+1,\cdots,n \tag{2.1-9}$$

（3）消元：

① 将第 i 行元素同除以 $a_{ii}^{(i-1)}$：

$$\begin{cases} a_{ik}^{(i)} = a_{ik}^{(i-1)} \big/ a_{ii}^{(i-1)} \\ b_p^{(i)} = b_i^{(i-1)} \big/ a_{ii}^{(i-1)} \end{cases} \qquad k = i, i+1, \cdots, n \tag{2.1-10}$$

② 消元：

$$\begin{cases} a_{jk}^{(i)} = a_{jk}^{(i-1)} - a_{ik}^{(i)} a_{ji}^{(i-1)} & j = i+1, i+2, \cdots, n \\ b_j^{(i)} = b_j^{(i-1)} - b_i^{(i)} a_{ji}^{(i-1)} & k = i+1, i+2, \cdots, n \end{cases} \tag{2.1-11}$$

2．回代

回代计算公式不变：

$$\begin{cases} x_n = b_n^{(n-1)} \big/ a_{nn}^{(n-1)} \\ x_i = b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j \end{cases} \qquad i = n-1, n-2, \cdots, 1 \tag{2.1-12}$$

用 Gauss 列主元素消元法就可求解例 3 和例 4 所给出的方程组.

对于例 3，

$$\begin{cases} x_2 = 1 \\ x_1 + 2x_2 = 3 \end{cases}$$

求解过程如下：

交换第一个方程和第二个方程得

$$\begin{cases} x_1 + 2x_2 = 3 \\ x_2 = 1 \end{cases}$$

回代求解得

$$\begin{cases} x_2 = 1 \\ x_1 = 1 \end{cases}$$

对于例 4，

$$\begin{cases} 10^{-20} x_1 + x_2 = 1 + 10^{-20} \\ x_1 + x_2 = 2 \end{cases}$$ （实际在内存里只有 1，没有 10^{-20}）

按 Gauss 列主元素消元法的计算过程如下：

交换第一个方程和第二个方程位置得

$$\begin{cases} x_1 + x_2 = 2 \\ 10^{-20} x_1 + x_2 = 1 + 10^{-20} \end{cases}$$

消元得

$$\begin{cases} x_1 + x_2 = 2 \\ x_2 = 1 \end{cases}$$

解之得

$$\begin{cases} x_2 = 1 \\ x_1 = 1 \end{cases}$$

对于例 2（仍用增广矩阵），计算过程如下.

第一轮消元：

$$\begin{bmatrix} 2 & 1 & 2 & 1 & 14 \\ 1 & 2 & -2 & 1 & 3 \\ 3 & 1 & 1 & 2 & 16 \\ 2 & 1 & 1 & 1 & 11 \end{bmatrix} \to \begin{bmatrix} 3 & 1 & 1 & 2 & 16 \\ 1 & 2 & -2 & 1 & 3 \\ 2 & 1 & 2 & 1 & 14 \\ 2 & 1 & 1 & 1 & 11 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1/3 & 1/3 & 2/3 & 16/3 \\ 1 & 2 & -2 & 1 & 3 \\ 2 & 1 & 2 & 1 & 14 \\ 2 & 1 & 1 & 1 & 11 \end{bmatrix} \to \begin{bmatrix} 1 & 1/3 & 1/3 & 2/3 & 16/3 \\ 0 & 5/3 & -7/5 & 1/3 & -7/5 \\ 0 & 1/3 & 4/3 & -1/3 & 10/3 \\ 0 & 1/3 & 1/3 & -1/3 & 1/3 \end{bmatrix}$$

第二轮消元：

$$\begin{bmatrix} 1 & 1/3 & 1/3 & 2/3 & 16/3 \\ 0 & 5/3 & -7/5 & 1/3 & -7/5 \\ 0 & 1/3 & 4/3 & -1/3 & 10/3 \\ 0 & 1/3 & 1/3 & -1/3 & 1/3 \end{bmatrix} \to \begin{bmatrix} 1 & 1/3 & 1/3 & 2/3 & 16/3 \\ 0 & 1 & -7/5 & 1/5 & -7/5 \\ 0 & 0 & 9/5 & -2/5 & 19/5 \\ 0 & 0 & 4/5 & -2/5 & 4/5 \end{bmatrix}$$

第三轮消元：

$$\begin{bmatrix} 1 & 1/3 & 1/3 & 2/3 & 16/3 \\ 0 & 1 & -7/5 & 1/5 & -7/5 \\ 0 & 0 & 9/5 & -2/5 & 19/5 \\ 0 & 0 & 4/5 & -2/5 & 4/5 \end{bmatrix} \to \begin{bmatrix} 1 & 1/3 & 1/3 & 2/3 & 16/3 \\ 0 & 1 & -7/5 & 1/5 & -7/5 \\ 0 & 0 & 1 & -2/9 & 19/9 \\ 0 & 0 & 0 & -2/9 & -8/9 \end{bmatrix}$$

解之得 $x_4 = 4$，利用式（2.1-12）得 $x_3 = 3$，$x_2 = 2$，$x_1 = 1$.

2.1.5 Gauss 全主元素消元法

若主元素不是在一列中选出的，而是在一个待消元的子矩阵中选出的，则称算法为全主元素消元法.

【定义 2】 $\left|a_{pq}\right| = \max\limits_{ij}\left|a_{ij}\right|$，$p = i, i+1, \cdots, n$，$q = i, i+1, \cdots, n$ 为第 i 轮消元的全主元素.

Gauss 全主元素消元法除了主元素是在一个子矩阵中而不是在一列中选出之外，还要进行列交换和行交换. 列交换后变量序号会发生变化，而输出时必须保留原序号不变，因此须增加一个记载变量原始序号的向量 $\boldsymbol{y}$.

Gauss 全主元素消元法的计算过程仍由消元和回代两部分组成，其计算过程和计算公式如下.

1．消元（共计 $n-1$ 轮）

第 i 轮消元过程如下（$i=1,2,\cdots,n-1$）.

① 找全主元素 a_{pq}（即确定全主元素所在行 p 与所在列 q）：

$$\begin{cases} p=i,\ q=i \\ p=j \\ q=k \end{cases} \qquad \left|a_{jk}^{(i-1)}\right| > \left|a_{pq}\right|, \quad \begin{matrix} j=i+1,i+2,\cdots,n \\ k=i+1,i+2,\cdots,n \end{matrix} \tag{2.1-13}$$

② 记下 x_i 和 x_q 的序号：

$$\begin{cases} y_i^{(0)}=i & i=1,2,\cdots,n \\ \begin{cases} y_i^{(i)}=q \\ y_q^{(i)}=i \end{cases} & i=1,2,\cdots,n-1 \end{cases} \tag{2.1-14}$$

③ 将第 i 行元素和第 p 行元素（含右端项）交换，将第 i 列元素和第 q 列元素交换：

$$\begin{cases} a_{pk}^{(i-1)} \Leftrightarrow a_{ik}^{(i-1)} & k=i,i+1,\cdots,n \\ b_p^{(i-1)} \Leftrightarrow b_i^{(i-1)} & \\ a_{jq}^{(i-1)} \Leftrightarrow a_{ji}^{(i-1)} & j=1,2,\cdots,n \end{cases} \tag{2.1-15}$$

④ 消元（和 Gauss 消元法一样）：

$$\begin{cases} a_{ik}^{(i)}=a_{ik}^{(i-1)}\big/a_{ii}^{(i-1)} & k=i+1,i+2,\cdots,n \\ b_i^{(i)}=b_i^{(i-1)}\big/a_{ii}^{(i-1)} & \\ a_{jk}^{(i)}=a_{jk}^{(i-1)}-a_{ik}^{(i-1)}a_{ji}^{(i-1)} & \begin{matrix} j=i+1,i+2,\cdots,n \\ k=i+1,i+2,\cdots,n \end{matrix} \\ b_j^{(i)}=b_j^{(i-1)}-b_i^{(i-1)}a_{ji}^{(i-1)} & \end{cases} \tag{2.1-16}$$

⑤ 计算 x_n 和回代求解：

$$\begin{cases} x_n=b_n^{(n-1)}\big/a_{nn}^{(n-1)} \\ x_i=b_i-\sum\limits_{j=i+1}^{n}a_{ij}^{(i)}x_j & i=n-1,n-2,\cdots,1 \end{cases} \tag{2.1-17}$$

⑥ 解序号还原：

$$x_i \Rightarrow x_{y_i} \qquad i=1,2,\cdots,n\,. \tag{2.1-18}$$

【例 5】 利用 Gauss 全主元消元法解方程组

$$\begin{cases} x_1+2x_2+3x_3=14 \\ 2x_1+x_2+4x_3=16 \\ x_1+x_2-x_3=0 \end{cases}$$

【解】 取 y_i 先存放第 i 个变量序号，$y_1=1$，$y_2=2$，$y_3=3$.

第一轮消元

（1）选全主元并将第二行与第一行交换，第一列和第三列交换，全主元素为 $a_{23}^{(0)}$：

$$\begin{bmatrix} 1 & 2 & 3 & 14 \\ 2 & 1 & 4 & 16 \\ 1 & 1 & -1 & 0 \end{bmatrix} \xrightarrow[\text{与第二行}]{\text{互换第一行}} \begin{bmatrix} 2 & 1 & 4 & 16 \\ 1 & 2 & 3 & 14 \\ 1 & 1 & -1 & 0 \end{bmatrix} \xrightarrow[\text{与第一列}]{\text{互换第三列}} \begin{bmatrix} 4 & 1 & 2 & 16 \\ 3 & 2 & 1 & 14 \\ -1 & 1 & 1 & 0 \end{bmatrix}$$

各列中变量原序号为 $y_1=3, y_2=2, y_3=1$.

（2）消元：

$$\begin{bmatrix} 1 & 0.25 & 0.5 & 4 \\ 3 & 2 & 1 & 14 \\ -1 & 1 & 1 & 0 \end{bmatrix} \to \begin{bmatrix} 1 & 0.25 & 0.5 & 4 \\ 0 & 1.25 & -0.5 & 2 \\ 0 & 1.25 & 1.5 & 4 \end{bmatrix}$$

第二轮消元

（1）选主元，全主元素为 $a_{33}^{(1)}=1.5$，$y_2=1$，$y_3=2$：

$$\begin{bmatrix} 1 & 0.25 & 0.5 & 4 \\ 0 & 1.25 & -0.5 & 2 \\ 0 & 1.25 & 1.5 & 4 \end{bmatrix} \xrightarrow[\text{与第三行}]{\text{互换第二行}} \begin{bmatrix} 1 & 0.25 & 0.5 & 4 \\ 0 & 1.25 & 1.5 & 4 \\ 0 & 1.25 & -0.5 & 2 \end{bmatrix} \xrightarrow[\text{与第二列}]{\text{互换第三列}} \begin{bmatrix} 1 & 0.5 & 0.25 & 4 \\ 0 & 1.5 & 1.25 & 4 \\ 0 & -0.5 & 1.25 & 2 \end{bmatrix}$$

（2）消元：

$$\begin{bmatrix} 1 & 0.5 & 0.25 & 4 \\ 0 & 1 & 5/6 & 8/3 \\ 0 & -0.5 & 1.25 & 2 \end{bmatrix} \to \begin{bmatrix} 1 & 0.5 & 0.25 & 4 \\ 0 & 1 & 5/6 & 8/3 \\ 0 & 0 & 20/12 & 20/6 \end{bmatrix}$$

2．回代

回代求解得

$$x_{y_3}=x_2=2，\quad x_{y_2}=x_1=1，\quad x_{y_1}=x_3=3.$$

2.1.6 Gauss 列主元法和 Gauss 全主元法计算量

Gauss 全主元素法求解方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 和 Gauss 消元法比较，只增加了找主元素及行列交换，其余完全相同，而找主元素的计算量级为 $\frac{1}{3}n^3$ 次选绝对值最大的元素，而数的交换次数不超过 n^2 次，这些计算都属于简单计算，因此 Gauss 全主元素消元法的计算量级和 Gauss 消元法一样都是 $\frac{1}{3}n^3$ 次乘法．Gauss 列主元素消元法的计算量级也为 $\frac{1}{3}n^3$ 次乘法．

2.1.7 Gauss 全主元素消元法计算程序

Gauss 消元法不是实用算法，Gauss 列主元素消元法和全主元素消元法才是实用算法，因篇幅关系，这里只附上 Gauss 全主元素消元法的计算程序．

```
#include "math.h"
#define  N  10

void main()
{
    int i, j, k, m, n, p, q, y[N+1];
    double a[N+1][N+1], b[N+1], s;

    printf("input n=");
    scanf("%d",&n);

    for(i=1;i<=n;i++)
        y[i]=i;
    printf("input b=");
    for(i=1;i<=n;i++)
        scanf("%lf",&b[i]);
    printf("input a=");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%lf",&a[i][j]);
    for(i=1;i<n;i++)
    {
        p=q=i;
        s=fabs(a[i][i]);
        for(j=i;j<=n;j++)
            for(k=i;k<=n;k++)
                if(fabs(a[j][k])>s)
                {
                    s=fabs(a[j][k]);
                    p=j;
                    q=k;
                }
        for(j=i;j<=n;j++)
        {
            s=a[i][j];
            a[i][j]=a[p][j];
            a[p][j]=s;
        }
        s=b[i];
        b[i]=b[p];
        b[p]=s;
        for(k=1;k<=n;k++)
        {
            s=a[k][i];
```

```
            a[k][i]=a[k][q];
            a[k][q]=s;
        }
        m=y[i];
        y[i]=y[q];
        y[q]=m;
        for(k=i+1;k<=n;k++)
            a[i][k]=a[i][k]/a[i][i];
        b[i]=b[i]/a[i][i];
        for(j=i+1;j<=n;j++)
        {
            for(k=i+1;k<=n;k++)
                a[j][k]=a[j][k]-a[i][k]*a[j][i];
            b[j]=b[j]-b[i]*a[j][i];
        }
    }
    b[n]=b[n]/a[n][n];
    for(i=n-1;i>=1;i--)
        for(j=i+1;j<=n;j++)
            b[i]=b[i]-a[i][j]*b[j];
    for(i=1;i<=n;i++)
        printf("x%d=%13.6f\n",y[i],b[i]);
}
```

使用例 5 的数据运行该程序，结果为

```
x[3]=3.000000,   x[1]=1.000000,   x[2]=2.000000
```

程序说明：

（1）对于 10 阶以上的方程组，由于输入量大，常将输入数据（系数矩阵和右端项）先存入数据文件中，程序运行时再读入相应变量，否则难以保证数据的正确性.

（2）从消元和回代算式可看出，在计算 $a_{ik}^{(i)}$ 时，用过 $a_{ik}^{(i-1)}$ 和 $b_i^{(i-1)}$ 后，$a_{ik}^{(i-1)}$ 和 $b_i^{(i-1)}$ 不再使用，同样计算 $a_{ik}^{(i)}$ 时，$a_{ik}^{(i-1)}$ 也只用一次，回代时 $b_i^{(i-1)}$ 也只用一次，因此程序中消元前后的系数矩阵都放在原系数矩阵中，右端项 $\boldsymbol{b}$ 在输入时为右端，消元时是变化后的右端项，回代时变成了解 x．以后这类情况不再说明.

2.1.8 消元法适用范围

【定义 3】矩阵 $\boldsymbol{A}$ 中前 $i(\leqslant n)$ 行 i 列所组成的矩阵称为 i 阶主子矩阵，主子矩阵所对应的行列式称为主子行列式.

要使 Gauss 消元法能进行到底，必须满足 $a_{ii}^{(i-1)} \neq 0$，$i=1,2,\cdots,n$．什么情况下 $a_{ii}^{(i-1)}$ 才不为零呢？下面的定理回答了这一问题.

【定理 1】Gauss 消元法能进行到底的充分必要条件是各阶主子行列式都不为零.

【证明】先证充分性.

设 $\boldsymbol{A}$ 的各阶主子行列式 $\det \boldsymbol{A}_k \neq 0$，$k=1,2,\cdots,n$．对于 Gauss 消元法所有运算只包含：

（1）将矩阵中的一行乘上一个数并与另一行相加；

（2）将矩阵某一行除以一个数 c_i．

第（1）种运算不改变所对应的行列式的值，第（2）种运算使行列式的值缩小 c_i 倍．若 Gauss 消元法的第 k 轮消元不能进行，则 k 阶主子行列式的值为 $a_{11}^{(1)}a_{22}^{(2)}\cdots a_{kk}^{(k)}$．当 $\det \boldsymbol{A}_k \neq 0$ 时，$a_{ii}^{(i)} \neq 0$，$i=1,2,\cdots,k$，前 k 轮消元能进行．当各阶主子行列式都不为零时，自然 $a_{11}^{(1)}a_{22}^{(2)}\cdots a_{mn}^{(n)}$ 不为 0，由此有 $a_{ii}^{(i)}$ 不为零，消元能进行．

能用 Gauss 消元法求解方程组，消元必须有意义，当 $\det \boldsymbol{A}_k = 0$ 时，$1 \leqslant k \leqslant n$，则 $a_{kk}^{(k)} = 0$，第 k 轮消元无法进行．

【定理 2】对于方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，只要 $\det \boldsymbol{A} \neq 0$，则可用 Gauss 列主元或全主元消元法求解．

证明略．

Gauss 消元法、列主元素法及全主元素法并不一定能得到满意解，本章最后一节将做进一步分析．

2.2　矩阵三角分解法

一个方阵可分解成两个以上矩阵的乘积，当给定一些约束后，这些分解将是唯一的．对矩阵做三角分解实际上只是将矩阵分解成两个三角形矩阵的乘积或两个三角形矩阵和一角对角线矩阵的乘积，分解的目的在于方便求解．

2.2.1　*LU* 分解法

【定义 1】对于 n 阶矩阵 $\boldsymbol{A}$，

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

若存在下三角方阵 $\boldsymbol{L}$ 和上三角方阵 $\boldsymbol{U}$，使得 $\boldsymbol{A}=\boldsymbol{LU}$，则称方阵 $\boldsymbol{A}$ 有三角分解或 $\boldsymbol{LU}$ 分解．特别地，若 $\boldsymbol{L}$ 为单位下三角矩阵，则称它为 Doolittle 分解，若 $\boldsymbol{U}$ 为单位上三角矩阵，则称它为 Grout 分解．本书只介绍第一种分解．

【定理 1】若 $\boldsymbol{A}$ 的各阶主子矩阵非奇异,则可进行 $\boldsymbol{LU}$ 分解．

证明方法和过程同上一节定理 1 的证明类似，证明略．

1．矩阵 *A* 的 Doolittle *LU* 分解

令

$$\boldsymbol{L}=\begin{bmatrix}1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \cdots & \cdots & \cdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn-1} & 1\end{bmatrix},\qquad \boldsymbol{U}=\begin{bmatrix}u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \cdots \\ & & & u_{nn}\end{bmatrix}$$

由 $\boldsymbol{A}=\boldsymbol{LU}$ 得

$$a_{ij}=\sum_{k=1}^{\min(i,j)} l_{ik}u_{kj}\ ,\quad i=1,2,\cdots,n\ ,\quad j=1,2,\cdots,n \tag{2.2-1}$$

2．*LU* 法分解步骤

方程组（2.2-1）共有 n^2 个未知数，分解步骤如下.

（1）计算 $\boldsymbol{U}$ 的第一行和 $\boldsymbol{L}$ 的第一列：

$$\begin{cases}u_{1j}=a_{1j}, & j=1,2,\cdots,n \\ \begin{cases}l_{11}=1 \\ l_{i1}=a_{i1}/u_{11}, & i=2,3,\cdots,n\end{cases}\end{cases} \tag{2.2-2}$$

（2）计算 $\boldsymbol{U}$ 的第 k 行和 $\boldsymbol{L}$ 的第 k 列：

$$\begin{cases}u_{kj}=a_{kj}-\sum_{p=1}^{k-1}l_{kp}u_{pj}, & k=1,2,\cdots,n;\ j=k,k+1,\cdots,n \\ l_{ik}=\left(a_{ik}-\sum_{p=1}^{k-1}l_{ip}u_{pj}\right)\Big/u_{kk}, & i=k+1,k+2,\cdots,n\end{cases} \tag{2.2-3}$$

2.2.2 *LU* 分解算例

【例 1】 求矩阵

$$\boldsymbol{A}=\begin{bmatrix}1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256\end{bmatrix}$$

的 $\boldsymbol{LU}$ 分解.

【解】 按式（2.2-2）得

$$u_{11}=1\ ,\quad u_{12}=2\ ,\quad u_{13}=3\ ,\quad u_{14}=4\ .$$
$$l_{11}=1\ ,\quad l_{21}=1\ ,\quad l_{31}=1\ ,\quad l_{41}=1\ .$$

按式（2.2-3）得

$$u_{22}=a_{22}-\sum_{p=1}^{1}l_{21}u_{12}=4-1\times 2=2\ .$$

$$u_{23}=a_{23}-l_{21}u_{13}=9-1\times 3=6\,.$$
$$u_{24}=a_{24}-l_{41}u_{14}=16-4=12\,.$$
$$l_{22}=1\,.$$
$$l_{32}=(a_{32}-l_{31}u_{12})/u_{22}=(8-1\times 2)/2=3\,.$$
$$l_{42}=(a_{42}-l_{41}u_{12})/u_{22}=(16-1\times 2)/2=7\,.$$
$$u_{33}=a_{33}-l_{31}u_{13}-l_{32}u_{23}=27-1\times 3-3\times 6=6\,.$$
$$u_{34}=a_{34}-l_{31}u_{14}-l_{32}u_{24}=64-1\times 4-3\times 12=24\,.$$
$$l_{33}=1\,.$$
$$l_{43}=(a_{43}-l_{41}u_{13}-l_{42}u_{23})/u_{33}=(81-1\times 3-7\times 6)/6=6\,.$$
$$u_{44}=a_{44}-l_{41}u_{14}-l_{42}u_{24}-l_{43}u_{34}=256-1\times 4-7\times 12-6\times 24=24\,.$$

即

$$\begin{bmatrix}1&2&3&4\\1&4&9&16\\1&8&27&64\\1&16&81&256\end{bmatrix}=\begin{bmatrix}1&&&\\1&1&&\\1&3&1&\\1&7&6&1\end{bmatrix}\times\begin{bmatrix}1&2&3&4\\&2&6&12\\&&6&24\\&&&24\end{bmatrix}.$$

2.2.3　利用 *LU* 分解法解方程组

利用 ***LU*** 分解法解方程组的求解公式和过程如下.

（1）利用式（2.2-2）和式（2.2-3）将方程组

$$\boldsymbol{Ax}=\boldsymbol{b}$$

转换成

$$\boldsymbol{LUx}=\boldsymbol{b}\,.$$

（2）令 $\boldsymbol{Ux}=\boldsymbol{y}$，则 $\boldsymbol{Ly}=\boldsymbol{b}$：

$$\begin{cases}y_1=b_1\\ y_i=b_i-\sum\limits_{j=1}^{i-1}l_{ij}y_j & i=2,3,\cdots,n\end{cases}\tag{2.2-4}$$

（3）求解 $\boldsymbol{Ux}=\boldsymbol{y}$：

$$\begin{cases}x_n=y_n/u_{nn}\\ x_i=\left(y_i-\sum\limits_{j=i+1}^{n}u_{ij}y_j\right)\Big/u_{ii} & i=n-1,n-2,\cdots,1\end{cases}\tag{2.2-5}$$

2.2.4　*LU* 分解法解方程组算例

【例 2】求解方程组

$$\begin{cases} x_1+2x_2+3x_3+4x_4=2 \\ x_1+4x_2+9x_3+16x_4=10 \\ x_1+8x_2+27x_3+64x_4=44 \\ x_1+16x_2+81x_3+256x_4=190 \end{cases}.$$

【解】由上例计算结果有

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}=\begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ 1 & 3 & 1 & \\ 1 & 7 & 6 & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & 3 & 4 \\ & 2 & 6 & 12 \\ & & 6 & 24 \\ & & & 24 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}=\begin{bmatrix} 2 \\ 10 \\ 44 \\ 190 \end{bmatrix}.$$

令

$$\boldsymbol{y}=\begin{bmatrix} 1 & 2 & 3 & 4 \\ & 2 & 6 & 12 \\ & & 6 & 24 \\ & & & 24 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

先求解

$$\begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ 1 & 3 & 1 & \\ 1 & 7 & 6 & 1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}=\begin{bmatrix} 2 \\ 10 \\ 44 \\ 190 \end{bmatrix}$$

得

$$\boldsymbol{y}=\begin{bmatrix} 2 \\ 8 \\ 18 \\ 24 \end{bmatrix}.$$

再求解

$$\boldsymbol{y}=\begin{bmatrix} 1 & 2 & 3 & 4 \\ & 2 & 6 & 12 \\ & & 6 & 24 \\ & & & 24 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}=\begin{bmatrix} 2 \\ 8 \\ 18 \\ 24 \end{bmatrix}$$

得

$$x_4=1,\ x_3=-1,\ x_2=1,\ x_1=-1.$$

LU 分解法解方程组的使用条件和 Gauss 消元法一样，***LU*** 分解无其他特性，因而更是一个实用性不强的算法.

2.2.5　平方根法和改进平方根法

1．平方根法和改进平方根法现实背景

大多数工程物理问题，尤其是力学问题，若要归结于一个线性代数方程组求解问题，那么这些方程组的系数矩阵通常具有以下特点：

（1）正定（对于力学问题通常是超静定的，静定对应正定）；

（2）对角严格占优；

（3）系数非零元素少，零元素多，且非零元素多在对角线两侧形成一个带形．这就是所谓的稀疏性和带形性．

【定义 2】若方阵 $\boldsymbol{A}$ 满足

（1） $\boldsymbol{A}=\boldsymbol{A}^{\mathrm{T}}$，

（2）对于任给向量 $\boldsymbol{x}\neq 0$，有 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}>0$，

则称矩阵 $\boldsymbol{A}$ 正定．

【定义 3】对于矩阵 $\boldsymbol{A}$，若 $|a_{ii}|\geqslant\sum_{j=1,j\neq i}^{n}|a_{ij}|$（$i=1,2,\cdots,n$），则称矩阵 $\boldsymbol{A}$ 行对角占优；若 $|a_{ii}|>\sum_{i=1,j\neq i}^{n}|a_{ij}|$（$j=1,2,\cdots,n$），则称矩阵 $\boldsymbol{A}$ 行对角严格占优，同样可定义列对角占优和列对角严格占优．对角占优也称弱对角占优．

【引理】若 $\boldsymbol{A}$ 是正定矩阵，则

（1） $\boldsymbol{A}$ 的所有主子矩阵 $\boldsymbol{A}_i$ 也是正定矩阵；

（2） $\boldsymbol{A}$ 非奇异， $\boldsymbol{A}^{-1}$ 也是正定矩阵；

（3） $a_{ii}>0\ (i=1,2,\cdots,n)$；

（4） $\boldsymbol{A}$ 的所有特征值都为正数；

（5） $\det\boldsymbol{A}_i>0$．

【证明】

（1）显然 $\boldsymbol{A}_i$ 也是对称矩阵，选 n 维向量 $\boldsymbol{y}=(x_1,x_2,\cdots,x_i,0,0,\cdots,0)^{\mathrm{T}}$，则由 $\boldsymbol{A}$ 是正定矩阵有

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}=\boldsymbol{y}^{\mathrm{T}}\boldsymbol{A}_i\boldsymbol{y}>0\,.$$

所以 $\boldsymbol{A}_i$ 也是正定矩阵．

（2）用反证法．若 $\det\boldsymbol{A}=0$，则存在 $\boldsymbol{x}\neq 0$，使得 $\boldsymbol{A}\boldsymbol{x}=0$，由此有 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}=0$，这与正定矩阵定义矛盾，所以 $\boldsymbol{A}$ 非奇异．又因 $(\boldsymbol{A}^{-1})^{\mathrm{T}}=(\boldsymbol{A}^{\mathrm{T}})^{-1}=\boldsymbol{A}^{-1}$，故 $\boldsymbol{A}^{-1}$ 也是对称矩阵，对任意非零向量 $\boldsymbol{y}\neq 0$，令 $\boldsymbol{x}=\boldsymbol{A}^{-1}\boldsymbol{y}$，则 $\boldsymbol{x}\neq 0$，从而有

$$\boldsymbol{y}^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{y}=(\boldsymbol{A}\boldsymbol{x})^{\mathrm{T}}\boldsymbol{A}^{-1}(\boldsymbol{A}\boldsymbol{x})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}>0\,,$$

故 $\boldsymbol{A}^{-1}$ 也是正定矩阵．

（3）令 $\boldsymbol{x}=\boldsymbol{e}_i$（$n$ 阶单位矩阵的第 i 列），则

$$0 < \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}_i = a_{ii}\,.$$

（4）设 λ 是 $\boldsymbol{A}$ 的任一特征值，$\boldsymbol{x}$ 是对应特征向量，则

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}}\lambda\boldsymbol{x} = \lambda\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} > 0\,,$$

又因 $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} > 0$，所以 $\lambda > 0$.

（5）设 $\lambda_1, \lambda_2, \cdots, \lambda_i$ 是 $\boldsymbol{A}_i$ 的特征值，则

$$\det \boldsymbol{A}_i = \lambda_1\lambda_2\cdots\lambda_i > 0\,.$$

2．平方根法计算公式和计算过程

对于线性方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$，若 $\boldsymbol{A}$ 是正定矩阵，则可用平方根法求其解，求解过程如下.

（1）三角分解

令 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$，$\boldsymbol{L}$ 为下三角矩阵，由 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ 得

$$a_{ij} = \sum_{k=1}^{j} l_{ik}l_{kj}^{T} = \sum_{k=1}^{j} l_{ik}l_{jk}, \qquad \begin{matrix} i = 1,2,\cdots,n \\ j = 1,2,\cdots,n \end{matrix} \tag{2.2-6}$$

由式（2.2-6）有

$$\begin{cases} l_{ij} = \left(a_{ij} - \sum\limits_{k=1}^{j-1} l_{ik}l_{jk}\right) \Big/ l_{jj}, & \begin{matrix} i = 1,2,\cdots,n \\ j = 1,2,\cdots,\ i-1 \end{matrix} \\ l_{ii} = \sqrt{a_{ii} - \sum\limits_{k=1}^{i-1} l_{ik}^2}\ , & i = 1,2,\cdots,n \end{cases} \tag{2.2-7}$$

（2）求解

令 $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{y}$，则

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}\,.$$

$$\begin{cases} y_1 = b_1 / l_{11} \\ y_i = \left(b_i - \sum\limits_{j=1}^{i-1} l_{ij}y_j\right) \Big/ l_{ii} & j = 2,3,\cdots,n \end{cases} \tag{2.2-8}$$

$$\begin{cases} x_n = y_n / l_{nn} \\ x_i = \left(y_i - \sum\limits_{j=i+1}^{n} l_{ij}y_j\right) \Big/ l_{ii} \\ \quad = \left(y_i - \sum\limits_{j=i+1}^{n} l_{ji}y_j\right) \Big/ l_{ii} & i = n-1, n-2, \cdots, 1 \end{cases} \tag{2.2-9}$$

3．平方根法计算实例及计算过程

【例 3】 用平方根法求解线性方程组

$$\begin{bmatrix}1 & 2 & 1\\ 2 & 5 & 0\\ 1 & 0 & 14\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}4\\ 7\\ 15\end{bmatrix}.$$

【解】

（1）用式（2.2-7）对 $\boldsymbol{A}$ 做三角分解.

$l_{11}=\sqrt{a_{11}}=\sqrt{1}=1$

$l_{21}=a_{21}/l_{11}=2$

$l_{22}=\sqrt{a_{22}-l_{21}^2}=\sqrt{5-2^2}=1$

$l_{31}=a_{31}/l_{11}=1/1=1$

$l_{32}=\left(a_{32}-l_{31}l_{21}\right)/l_{22}=\left(0-1\times 2\right)/1=-2$

$l_{33}=\sqrt{a_{33}-l_{31}^2-l_{32}^2}=\sqrt{14-1-4}=3$

即

$$\begin{bmatrix}1 & & \\ 2 & 1 & \\ 1 & -2 & 3\end{bmatrix}\begin{bmatrix}1 & 2 & 1\\ & 1 & -2\\ & & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}4\\ 7\\ 15\end{bmatrix}$$

（2）用式（2.2-8）、式（2.2-9）求解.

令

$$\begin{bmatrix}1 & 2 & 1\\ & 1 & -2\\ & & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}y_1\\ y_2\\ y_3\end{bmatrix}$$

则有

$$\begin{bmatrix}1 & & \\ 2 & 1 & \\ 1 & -2 & 3\end{bmatrix}\begin{bmatrix}y_1\\ y_2\\ y_3\end{bmatrix}=\begin{bmatrix}4\\ 7\\ 15\end{bmatrix}$$

解之得 $y_1=4$，$y_2=-1$，$y_3=3$.

解 $\begin{bmatrix}1 & 2 & 1\\ & 1 & -2\\ & & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}4\\ -1\\ 3\end{bmatrix}$ 得 $x_3=1$，$x_2=1$，$x_1=1$.

2.2.6　改进平方根法

平方根法要用到开方运算，为了避免开方运算可用改进平方根法.

1．改进平方根法计算公式和计算过程

对于方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，改进平方根法计算过程如下.

（1）三角分解

令 $\boldsymbol{A}=\boldsymbol{LDL}^{\mathrm{T}}$，$\boldsymbol{L}$ 为下三角单位矩阵，$\boldsymbol{D}$ 是对角线矩阵：

$$\boldsymbol{L}=\begin{bmatrix}1 & & & \\ l_{21} & 1 & & \\ \cdots & \cdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1\end{bmatrix},\quad \boldsymbol{D}=\begin{bmatrix}d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n\end{bmatrix},\quad \boldsymbol{LD}=\begin{bmatrix}d_1 & & & \\ l_{21}d_1 & d_2 & & \\ \cdots & \cdots & \ddots & \\ l_{n1}d_1 & l_{n2}d_2 & \cdots & d_n\end{bmatrix}$$

按矩阵乘法公式有

$$a_{ij}=\sum_{k=1}^{j}l_{ik}d_k l_{jk}\ ,\quad i=1,2,\cdots,n\ ,\quad j=1,2,\cdots,n$$

由此得

$$\begin{cases}l_{ij}=\left(a_{ij}-\sum\limits_{k=1}^{j-1}l_{ik}d_k l_{jk}\right)\Big/d_j \\ d_i=a_{ii}-\sum\limits_{k=1}^{i-1}l_{ik}^2 d_k\end{cases},\quad \begin{matrix}i=1,2,\cdots,n\\ j=1,2,\cdots,i\end{matrix} \tag{2.2-10}$$

用式（2.2-10）所表述的改进平方根法计算量较大.

令 $\gamma_j^{(i)}=l_{ij}d_j$，$j=1,2,\cdots,i$，则

$$\gamma_j^{(i)}=a_{ij}-\sum_{k=1}^{j-1}\gamma_k^{(i)}l_{jk}\ ,\quad \begin{matrix}i=1,2,\cdots,n\\ j=1,2,\cdots,i\end{matrix} \tag{2.2-11}$$

$$\begin{cases}d_i=\gamma_i^{(i)} \\ l_{ij}=\gamma_i^{(i)}\big/d_j\end{cases},\quad \begin{matrix}i=1,2,\cdots,n\\ j=1,2,\cdots,i-1\end{matrix} \tag{2.2-12}$$

（2）求解

令 $\boldsymbol{DL}^{\mathrm{T}}\boldsymbol{x}=\boldsymbol{y}$，则

$$\boldsymbol{Ly}=\boldsymbol{b}\ .$$

$$\begin{cases}y_1=b_1 \\ y_i=b_i-\sum\limits_{j=1}^{i-1}l_{ij}y_j\end{cases}\qquad j=2,3,\cdots,n \tag{2.2-13}$$

令 $\boldsymbol{Lx}=\boldsymbol{z}$，则

$$z_i=y_i\big/d_i\ ,\quad i=1,2,\cdots,n \tag{2.2-14}$$

$$\begin{cases}x_n=z_n \\ x_i=z_i-\sum\limits_{j=i+1}^{n}l_{ji}x_j\end{cases}\qquad i=n-1,n-2,\cdots,1 \tag{2.2-15}$$

2．改进平方根法算例

【例 4】 用改进平方根法求解例 3 的方程组.

【解】

（1）用式（2.2-12）和式（2.2-13）做矩阵分解：

$d_1 = a_{11} = 1$

$\gamma_1^{(2)} = a_{21} = 2$

$l_{21} = \gamma_1^{(2)} / d_1 = 2$

$d_2 = \gamma_2^{(2)} = a_{22} - \gamma_1^{(2)} l_{21} = 1$

$\gamma_1^{(3)} = a_{31} = 1$

$l_{31} = \gamma_1^{(3)} / d_1 = 1$

$\gamma_2^{(3)} = a_{32} - \gamma_1^{(3)} a_{21} = 0 - 2 = -2$

$l_{32} = \gamma_2^{3} / d_2 = -2$

$\gamma_3^{(3)} = a_{33} - \gamma_1^{(3)} l_{31} - \gamma_2^{(3)} l_{32} = 14 - 1 - 4 = 9$

$d_3 = \gamma_3^{(3)} = 9$

分解得

$$\boldsymbol{A} = \begin{bmatrix} 1 & & \\ 2 & 1 & \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 9 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ & 1 & -2 \\ & & 1 \end{bmatrix}$$

（2）代入式（2.2-13）、式（2.2-14）和式（2.2-15）求解：

$y_1 = 4$，$y_2 = -1$，$y_3 = 9$

$z_1 = 4$，$z_2 = -1$，$z_3 = 1$

$x_3 = 1$，$x_2 = 1$，$x_1 = 1$

2.2.7　*LU* 分解法、平方根法和改进平方根法计算量

1．*LU* 分解法计算量

LU 分解法由 ***LU*** 分解和求解两部分组成.

（1）***LU*** 分解计算量

L 形成计算量：

① 除法 $n-1$ 次；

② 第 i 行乘法计算量：$1 = 2 + 3 + \cdots + (i-2) \doteq \frac{1}{2} i^2$；

因此，***L*** 形成计算量为 $\frac{1}{6} n^3$ 次乘法.

U 形成计算量：

第 i 列计算量：$1 = 2 + 3 + \cdots + (i-1) \doteq \frac{1}{2} i^2$；

因此，$\boldsymbol{U}$ 形成计算量为 $\frac{1}{6}n^3$ 次乘法.

综上，$\boldsymbol{LU}$ 分解计算量为 $\frac{1}{3}n^3$ 次乘法.

（2）求解计算量

很容易算出求解计算量为 n^2 次乘法，所以 $\boldsymbol{LU}$ 分解法计算量为 $\frac{1}{3}n^3$ 次乘法.

2．平方根法计算量

（1）三角分解计算量

由式（2.2-7）知计算第 i 行 l_{ij} 的乘法计算量为

$$1+2+\cdots+i-1=\frac{1}{2}(i-1)i \doteq \frac{1}{2}i^2$$

另外还要开方一次，由此知平方根法三角分解的计算量为 $\sum_{i=1}^{n}\frac{1}{2}i^2 \doteq \frac{1}{6}n^3$ 次乘法和 n 次开方．当 n 较大时，开 n 次方所用时间远比 $\frac{1}{3}n^3$ 次乘法所用时间少，所以平方根法三角分解所用计算量为 $\frac{1}{6}n^3$ 乘法.

（2）求解计算量

由式（2.2-8）和式（2.2-9）知求解所用乘（除）法计算量为

$$2\sum_{i=1}^{n} i \doteq n^2$$

次乘法．所以平方根法总的计算量为 $\frac{1}{6}n^3$ 次乘法.

3．改进平方根法计算量

（1）三角分解计算量

由式（2.2-11）和式（2.2-12）知，改进平方根法三角分解计算量为

$$\sum_{i=1}^{n}\frac{1}{2}i^2 \doteq \frac{1}{6}n^3$$

次乘法.

（2）求解计算量

由式（2.2-13）、式（2.2-14）和式（2.2-15）知求解所用乘（除）法计算量为

$$2\sum_{i=1}^{n} i \doteq n^2$$

次乘法和 n 次除法．所以改进平方根法总的计算量也为 $\frac{1}{6}n^3$ 次乘法.

2.2.8　变带宽压缩存储平方根法

通常认为 500 阶以下线性方程组为中、小型线性方程组，但用前面介绍的六种算法解 200 阶线性方程组都很难，原因是 200 阶线性方程组单系数矩阵就要至少占用 16 万字节的内存空间，现在的一般计算机的内存容量虽然都超过 512 MB，大于 16 万字节，不过受寻址方法限制只能访问内存中的部分字节，难以求解 200 阶线性方程组.

1．平方根法和改进平方根法算法特点

前面已经提到工程物理问题所归结的代数方程组的特性，上面介绍的六个算法最多只利用其对称性（平方根法和改进平方根法），实际上从式（2.2-8）～式（2.2-10）或从式（2.2-12）～式（2.2-15）可看出，若系数矩阵中第 i 行左起第一个非零元素为 a_{ij}，则经过三角分解之后，$\boldsymbol{L}$ 矩阵中第 i 行左起第一个非零元素的列号仍为 j，且求解时第 i 行第 j 列前的零元素也不起作用，因而可不存储（当然也不使用这些零元素）. 从而可大大降低内存开销，减少运行时间. 正因为这样，人们给出了变带宽压缩存储平方根法和变带宽压缩存储改进平方根法. 鉴于改进平方根法仅比平方根法只少了 n 次开方运算，而用现代计算机开 10000 次平方耗时不超过一分钟，但程序设计难度比后者更大，为了节省篇幅，加上改进平方根法在文献中易于查找，本书只介绍变带宽压缩存储平方根法.

2．变带宽压缩存储平方根法计算公式

【定义 4】 $\boldsymbol{A}$ 矩阵中第 i 行左起第一个非零元素到对角线元素的元素个数称为第 i 行的（半）带宽，并用 m_i 表示.

用 m_i 表示矩阵 $\boldsymbol{A}$ 的第 i 行的带宽，则第 i 行对角线元素序号可用下式计算：

$$\begin{cases} p_i = \sum_{j=1}^{i} m_j = p_{i-1} + m_i & i = 2,3,\cdots,n \\ p_1 = m_1 = 1 \end{cases} \tag{2.2-16}$$

第 i 行带上第 1 个非零元素的列号为

$$ij = i - m_i + 1 .$$

第 j 行带上第 1 个非零元素的列号为

$$jj = j - m_j + 1 .$$

第 i 行第 j 列的带上非零元素的序号为

$$q = p_{i-1} + m_i + j - i \qquad \begin{matrix} i = 1,2,\cdots,n \\ j = ij, ij+1,\cdots,i \end{matrix}$$

变带宽压缩存储平方根法计算过程如下.

（1）三角分解

根据矩阵乘法法则，令 $m_{ij} = \max(ij, jj)$，则有

$$a_q=\sum_{k=m_{ij}}^{j} l_{p_{i-1}+m_i+k-i}\,l_{p_{j-1}+m_j+k-j}\qquad \begin{matrix} i=1,2,\cdots,n \\ j=ij,ij+1,\cdots,i \end{matrix} \tag{2.2-17}$$

由式（2.2-17）得三角矩阵分解公式为

$$\begin{cases} l_q=\left(a_q-\sum\limits_{k=m_{ij}}^{j-1} l_{p_{i-1}+m_i+k-i}\,l_{p_{j-1}+m_j+k-j}\right)\Big/ l_{p_j} & i=1,2,\cdots,n \\ l_{p_i}=\left(a_{p_i}-\sum\limits_{k=ij}^{i-1} l_{p_{i-1}+m_i+k-i}^2\right)^{1/2} & j=ij,ij+1,\cdots,i \end{cases} \tag{2.2-18}$$

（2）求解

同样令

$$\begin{cases} \boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\boldsymbol{y} \\ \boldsymbol{L}\boldsymbol{y}=\boldsymbol{b} \end{cases}$$

求解公式为

$$\begin{cases} y_1=b_1/l_1 \\ y_i=\left(b_i-\sum\limits_{j=ij}^{i-1} l_{p_{i-1}+m_i-i+j}\,y_j\right)\Big/ l_{p_i} & i=2,3,\cdots,n \end{cases} \tag{2.2-19}$$

$$\begin{cases} x_n=y_n/l_{p_n} \\ x_i=\left(y_i-\sum\limits_{\substack{j=i+1 \\ jj\leqslant i}}^{n} l_{p_{j-1}+m_j-j+i}\,x_j\right)\Big/ l_{p_i} \end{cases} \tag{2.2-20}$$

压缩存储平方根法程序中只存放了 $\boldsymbol{L}$ 矩阵中的带上元素，式（2.2-20）中要用到 $\boldsymbol{L}^{\mathrm{T}}$ 矩阵中的元素．在 $\boldsymbol{L}^{\mathrm{T}}$ 中，$l_{ij}^{\mathrm{T}}=l_{ji}$，当 i 小于第 j 行带上第 1 个元素序号时，$\boldsymbol{L}$ 中无该元素．

3．变带宽压缩存储平方根法算例

【例 5】 用变带宽压缩存储平方根法求解方程组

$$\begin{bmatrix} 5 & 1 & 1 & & & & & & \\ 1 & 6 & 3 & 1 & & 1 & & & \\ 1 & 3 & 6 & 1 & & & & & \\ & 1 & 1 & 8 & 1 & & & & \\ & & & 1 & 9 & 2 & & & \\ & 1 & & & 2 & 10 & 1 & 1 & \\ & & & & & 1 & 8 & 1 & 2 \\ & & & & & 1 & 1 & 5 & 2 \\ & & & & & & 2 & 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} 10 \\ 32 \\ 29 \\ 45 \\ 61 \\ 87 \\ 88 \\ 71 \\ 75 \end{bmatrix}$$

九阶线性方程组求解用手算太烦琐，这里直接给出 C 程序.

```
#include "math.h"
#define N 9
main(){
    float a[]={0,5,1,6,1,3,6,1,1,8,1,9,1,0,0,2,10,1,8,1,1,5,2,2,5};
    float b[]={0,10,32,29,45,61,87,88,71,75};
    int m[]={0,1,2,3,3,2,5,2,3,3};
    int i, j, k, q, mij, ij, jj, p[N+1];
    p[1]=1;
    for(i=2;i<=N;i++)
       p[i]=p[i-1]+m[i];
    /*下面是三角分解*/
    for(i=2;i<=N;i++){
       for(k=1;k<=m[i];k++){
          j=i-m[i]+k;
          mij=max(i-m[i]+1,j-m[j]+1);
          for(q=mij;q<j;q++){
             iq=[p[i-1]+m[i]-i+q;
             jq=p[j-1]+m[j]-j+q;
             L[p[i-1]+k]=L[p[i-1]+k]-L[iq]*L[jq];
          }
          if(k!=m[i])
             L[p[i-1]+k]=L[p[i-1]+k]/L[p[j]];
          else
             L[p[i-1]+k]=sqrt(L[p[i]]);
       }
    }
    /*下面求解 yi*/
    b[1]=b[1]/L[1];
    for(i=2;i<=N;i++){
       for(j=i-m[i]+1;j<i;j++)
          b[i]=b[i]-L[p[i-1]+m[i]-i+j]*b[j];
       b[i]=b[i]/L[p[i]];
    }
    /*下面求解 xi*/
    b[N]=b[N]/L[p[N]];
    for(i=2;i>=1;i--){
       for(j=i+1;j<=N;j++)
          if(j-m[j]+1<=i)
             b[i]=b[i]-L[p[j-1]+m[j]-j+i]*x[j];
       b[i]=b[i]/a[p[i]];
    }
    /*输出结果*/
    for(i=1;i<=N;i++)
       printf("x%d=%f, ",i,b[i]);
}
```

程序运行结果为

```
x[1]=1.018600,x[2]=1.951031,x[3]=2.955970,x[4]=4.392488,x[5]=4.953098
x[6]=6.014817,x[7]=6.998037,x[8]=7.996566,x[9]=9.002159
```

变带宽压缩存储平方根法算法复杂，按计算程序复杂程度算法——环数数，程序复杂程度并不高，但程序设计难度大，若将算式表述成书中各公式，给出各式的计算过程和计算条件，程序设计难度就降低了.

为了节省篇幅，这里不再介绍变带宽压缩改进平方根法.

2.2.9 追赶法

1. 追赶法三角分解

追赶法用于求解三对角方程组. 由于算法求解过程类似追和赶，故称为追赶法. 追赶法属特殊的 $\boldsymbol{LU}$ 法，它有一定的实用价值.

【定义 5】若矩阵中只有主对角线和两条次对角线中是非零元素，则称矩阵为三对角矩阵.

三对角矩阵形为

$$\boldsymbol{A}=\begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{bmatrix}$$

【定理】若 $\boldsymbol{A}$ 是三对角矩阵，且

（1）$|b_i|\geqslant|a_i|+|c_i|$，$i=2,3,\cdots,n-1$；

（2）$|b_1|>|c_1|$，$|b_n|>|a_n|$.

则矩阵 $\boldsymbol{A}$ 非奇异，有

$$\begin{cases} \boldsymbol{A}=\boldsymbol{PQ}=\begin{bmatrix} p_1 & & & & \\ a_2 & p_2 & & & \\ & a_3 & p_3 & & \\ & & \ddots & \ddots & \\ & & & a_n & p_n \end{bmatrix}\times\begin{bmatrix} 1 & q_1 & & & \\ & 1 & q_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & q_{n-1} \\ & & & & 1 \end{bmatrix} \\ p_1=b_1 \\ q_i=\dfrac{c_i}{p_i} \\ p_i=b_i-a_iq_{i-1} \end{cases} \tag{2.2-21}$$

证明略.

这是一充分性定理，上面的分解称为 Grout 分解，同样可按 Doolittle 分解.

2．追赶法计算过程和计算实例

解对角方程组的计算过程仍由三角分解和求解两部分组成.

对于三对角方程组 $\boldsymbol{Ax}=\boldsymbol{d}$，求解步骤如下.

（1）三解分解

计算公式见式（2.2-21）.

（2）求解

令 $\boldsymbol{y}=\boldsymbol{Qx}$，则

$$\begin{cases} y_1=\dfrac{d_1}{p_1} \\ y_i=\dfrac{d_i-a_iy_{i-1}}{p_i} \qquad i=2,3,\cdots,n \end{cases} \tag{2.2-22}$$

$$\begin{cases} x_n=y_n \\ x_i=y_i-q_ix_{i+1} \qquad i=n-1,n-2,\cdots,1 \end{cases} \tag{2.2-23}$$

式（2.2-22）像追，y_i 的序号由小到大；式（2.2-23）似赶，x_i 的序号由大到小.

【例 6】 用追赶法解方程组

$$\begin{cases} 2x_1-x_2=1 \\ -x_1+2x_2-x_3=0 \\ -x_2+2x_3-x_4=0 \\ -x_3+2x_4=-1 \end{cases}.$$

【解】 按式（2.2-24）得

$$\boldsymbol{P}=\begin{bmatrix} 2 & & & \\ -1 & 3/2 & & \\ & -1 & 4/3 & \\ & & -1 & 5/4 \end{bmatrix},\quad \boldsymbol{Q}=\begin{bmatrix} 1 & -1/2 & & \\ & 1 & -2/3 & \\ & & 1 & -3/4 \\ & & & 1 \end{bmatrix}.$$

按式（2.2-23）得

$$\boldsymbol{y}=\begin{bmatrix} \dfrac{1}{2} & \dfrac{1}{3} & \dfrac{1}{4} & 1 \end{bmatrix}^{\mathrm{T}}.$$

按式（2.2-24）得

$$\boldsymbol{x}=[1 \quad 1 \quad 1 \quad 1]^{\mathrm{T}}.$$

本章介绍的几个算法以 Gauss 列主元素消元法和全主元素消元法要求的条件最低，这两个算法的复杂性和 Gauss 消元法以及 $\boldsymbol{LU}$ 分解法一样．因而 Gauss 消元法和 $\boldsymbol{LU}$ 分解法的实用性不强．平方根法和改进平方根法要求的条件更严，这两个算法之所以实用性强，原因是可以压缩存储，从而算法的时间和空间复杂性都可大大降低，而实际工程物理问题中方程组性质本身就良好，能满足算法要求（$\boldsymbol{LU}$ 法也可压缩存储，但无法利用对称性）.

2.3 范数简介

矩阵和向量无大小之分，为了表述矩阵和向量的性质，须引入范数，范数用 $\|*\|_p$ 表示，*可以是向量也可以是矩阵.

2.3.1 向量范数定义

向量范数必须满足

（1）$\|\boldsymbol{x}\| \geqslant 0$，只有 $\boldsymbol{x}=\boldsymbol{0}$ 时，$\|\boldsymbol{x}\|=0$；（范数的非负性）

（2）$\|\alpha \boldsymbol{x}\|=|\alpha|\|\boldsymbol{x}\|$；（范数的线性性）

（3）$\|\boldsymbol{x}+\boldsymbol{y}\| \geqslant \|\boldsymbol{x}\|+\|\boldsymbol{y}\|$.（三角不等性）

满足以上三个性质都可成为范数，这三个性质也可作为范数的定义.

2.3.2 常用向量范数

满足向量范数定义的范数不少，常用范数只有如下三个：

（1）1-范数：$\|\boldsymbol{x}\|_1=\sum_{i=1}^{n}|x_i|$；

（2）2-范数：$\|\boldsymbol{x}\|_2=\sqrt{\sum_{i=1}^{n}x_i^2}$；

（3）∞-范数：$\|\boldsymbol{x}\|_\infty=\max_i|x_i|$.

上述三个范数又称为列范数、谱范数和行范数. 显然这三个范数满足范数的定义，上述三个范数可统一表述成

$$\|\boldsymbol{x}\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{1/p}.$$

当 p 为 ∞ 时，有

$$\|\boldsymbol{x}\|_p=\lim_{p\to\infty}\left(\sum_{i=1}^{n}|x_i|^p\right)^{1/p}=\lim_{p\to\infty}\max_i(|x_i|^p)^{1/p}=\max_i|x_i|.$$

上述三个范数的名称在向量范数中意义不明显，但在矩阵范数中意义就清楚了.

2.3.3 向量范数性质

【性质 1】向量 $\boldsymbol{x}\in\boldsymbol{R}^n$ 的范数 $\|\boldsymbol{x}\|$ 是其分量的一致连续函数，即对任给 $\varepsilon>0$，存在正数 δ，对于 $\boldsymbol{h}\in\boldsymbol{R}^n$，当 $\max_i|h_i|<\delta$ 时有

$$\|\|\boldsymbol{x}+\boldsymbol{h}\|-\|\boldsymbol{x}\|\|<\varepsilon .$$

【证明】因为

$$\boldsymbol{h}=h_1\boldsymbol{e}_1+h_2\boldsymbol{e}_2+\cdots+h_n\boldsymbol{e}_n ,$$

其中 $\boldsymbol{e}_i$ 是 n 阶单位矩阵的 i 列.

$$\begin{aligned}\|\|\boldsymbol{x}+\boldsymbol{h}\|-\|\boldsymbol{x}\|\| &\leqslant \|\boldsymbol{h}\|=\|h_1\boldsymbol{e}_1+h_2\boldsymbol{e}_2+\cdots+h_n\boldsymbol{e}_n\| \\ &\leqslant \|h_1\boldsymbol{e}_1\|+\|h_2\boldsymbol{e}_2\|+\cdots+\|h_n\boldsymbol{e}_n\| \\ &\leqslant \max_i|h_i|\max_i\|\boldsymbol{e}_i\|\end{aligned}$$

令 $m=\max\limits_i\|\boldsymbol{e}_i\|$，取 $\delta=\varepsilon/m$，当 $\max\limits_i\|h_i\|<\delta$ 时有

$$\|\|\boldsymbol{x}+\boldsymbol{h}\|-\|\boldsymbol{x}\|\|<\varepsilon .$$

【性质 2】在 $\boldsymbol{R}^n$ 上定义的任意一种范数 $\|\boldsymbol{x}\|$ 都与 $\|\boldsymbol{x}\|_\infty$ 等价，即必存在与 $\boldsymbol{x}$ 无关的正数 m 和 M，$m\leqslant M$，使得对于一切非零向量 $\boldsymbol{x}\in\boldsymbol{R}^n$ 有

$$0<m\|\boldsymbol{x}\|_\infty\leqslant\|\boldsymbol{x}\|\leqslant M\|\boldsymbol{x}\|_\infty .$$

【证明】设 $\boldsymbol{y}\in\boldsymbol{R}^n,\boldsymbol{y}\neq\boldsymbol{0}$，由于 $\|\boldsymbol{y}\|$ 是其分量 $y_i(i=1,2,\cdots,n)$ 的一致连续函数，在有界闭集

$$\boldsymbol{S}=\{\boldsymbol{y}\in\boldsymbol{R}^n\mid\|\boldsymbol{y}\|_\infty=1\}$$

上一定存在最小值 $m>0$ 和最大值 $M\geqslant m$，于是对于任意 $\boldsymbol{y}\in\boldsymbol{S}$ 有

$$0<m\leqslant\|\boldsymbol{y}\|\leqslant M .$$

所以对于任意非零向量 $\boldsymbol{x}\in\boldsymbol{R}^n$，都有 $\dfrac{\boldsymbol{x}}{\|\boldsymbol{x}\|_\infty}\in\boldsymbol{S}$，因此

$$0<m\leqslant\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_\infty}\leqslant M ,$$

即

$$0<m\|\boldsymbol{x}\|_\infty\leqslant\|\boldsymbol{x}\|\leqslant M\|\boldsymbol{x}\|_\infty .$$

【推论】在 $\boldsymbol{R}^n$ 上定义的任何两种向量范数都是等价的.

范数的等价性表明，对于任意向量，如果它的某一种范数很大或很小，则其余范数也很大或很小．究竟采用何种范数，应考虑问题的特点和处理方便性.

2.3.4　矩阵范数定义

矩阵范数可由向量范数导出，在定义矩阵范数之前先讨论两个向量范数 $\|\boldsymbol{A}\boldsymbol{x}\|$ 和 $\|\boldsymbol{x}\|$ 之间的关系.

【定理 1】对于任意向量 $\boldsymbol{x}\in\boldsymbol{R}^n$，必存在一个与 $\boldsymbol{x}$ 无关的常数 C，使得

$$\|\boldsymbol{A}\boldsymbol{x}\|\leqslant C\|\boldsymbol{x}\| .$$

【证明】 由向量范数的连续性知$\|\boldsymbol{Ax}\|$是其分量$\sum_{j=1}^{n} a_{ij}x_j$的连续函数，因此在有界闭集

$$\boldsymbol{S}=\{\boldsymbol{x}\in\boldsymbol{R}^n \mid \|\boldsymbol{x}\|_\infty=1\}$$

上有上界D，对任意$\boldsymbol{x}\in\boldsymbol{R}^n$，$\dfrac{\boldsymbol{x}}{\|\boldsymbol{x}\|_\infty}\in\boldsymbol{S}$，故

$$\left\|\boldsymbol{A}\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_\infty}\right\|\leqslant D,$$

即

$$\|\boldsymbol{Ax}\|\leqslant D\|\boldsymbol{x}\|_\infty .$$

由向量范数的等价性可知，存在与向量$\boldsymbol{x}$无关的常数$m>0$，使得

$$0<m\|\boldsymbol{x}\|_\infty\leqslant\|\boldsymbol{x}\| .$$

所以

$$\|\boldsymbol{Ax}\|\leqslant D\frac{\|\boldsymbol{x}\|}{m}=C\|\boldsymbol{x}\| .$$

这一定理给出了可由向量范数定义矩阵范数的依据.

【定义 1】 对于任意矩阵$\boldsymbol{A}\in\boldsymbol{R}^{n\times n}$，用$\dfrac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|}$的上确界表示$\boldsymbol{A}$的范数，记为

$$\|\boldsymbol{A}\|=\sup\frac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|} .$$

由于$\dfrac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|}=\left\|\boldsymbol{A}\dfrac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right\|$，$\left\|\dfrac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\right\|=1$，所以定义 1 又可表述成

$$\|\boldsymbol{A}\|=\sup_{\|\boldsymbol{x}\|=1}\|\boldsymbol{Ax}\| .$$

又因为连续函数$\|\boldsymbol{Ax}\|$在有界闭集$\boldsymbol{S}=\{\boldsymbol{x}\in\boldsymbol{R}^n \mid \|\boldsymbol{x}\|=1\}$上必然可达到最大值，所以$\boldsymbol{A}$的范数又可定义成

$$\|\boldsymbol{A}\|=\max_{\|\boldsymbol{x}\|=1}\|\boldsymbol{Ax}\| .$$

2.3.5 矩阵范数基本性质

由向量范数的基本性质和矩阵范数的定义可直接得到矩阵范数的基本性质.

（1）$\|\boldsymbol{A}\|\geqslant 0$，仅当$\boldsymbol{A}=\boldsymbol{0}$时有$\|\boldsymbol{A}\|=0$；

（2）$\|\alpha\boldsymbol{A}\|=|\alpha|\|\boldsymbol{A}\|$；

（3）$\|\boldsymbol{A}+\boldsymbol{B}\|\leqslant\|\boldsymbol{A}\|+\|\boldsymbol{B}\|$；

（4）$\|\boldsymbol{Ax}\|\leqslant\|\boldsymbol{A}\|\|\boldsymbol{x}\|$；

（5）$\|\boldsymbol{AB}\| \leqslant \|\boldsymbol{A}\|\|\boldsymbol{B}\|$.

【定理 2】 $\|\boldsymbol{A}\|_1 = \max\limits_j \sum\limits_{i=1}^{n} |a_{ij}|$，$j = 1,2,\cdots,n$.

【证明】 $\|\boldsymbol{Ax}\|_1 = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}|a_{ij}x_j| \leqslant \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}|a_{ij}||x_j| = \sum\limits_{i=1}^{n}|x_j|\sum\limits_{j=1}^{n}|a_{ij}| \leqslant \sum\limits_{i=1}^{n}|x_j|\max\limits_j\sum\limits_{i=1}^{n}|a_{ij}|$

设 $\max\limits_j \sum\limits_{i=1}^{n}|a_{ij}| = \sum\limits_{i=1}^{n}|a_{ip}|$，令

$$x_i = \begin{cases} 0 & i \neq p \\ 1 & i = p \end{cases}，\quad \|\boldsymbol{x}\|_1 = 1$$

$$\|\boldsymbol{Ax}\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}x_j| = \sum_{i=1}^{n}a_{ip} = \max_j\sum_{i=1}^{n}|a_{ij}| = \|\boldsymbol{A}\|_1$$

【定理 3】 $\|\boldsymbol{Ax}\|_\infty = \max\limits_i \sum\limits_{j=1}^{n}|a_{ij}|$.

证明方法和定理 2 的证明类似，证明略.

【定理 4】 $\|\boldsymbol{A}\|_2 = \max\limits_i(\lambda_{\boldsymbol{AA}^{\mathrm{T}}})^{1/2}$，即 $\|\boldsymbol{A}\|_2$ 为 $\boldsymbol{AA}^{\mathrm{T}}$ 的最大特征值的平方根.

【证明】 由代数知识知 $\boldsymbol{AA}^{\mathrm{T}}$ 是非负定矩阵，其特征值均不为负数，记 $\lambda_1 < \lambda_2 < \cdots < \lambda_n$，且 $\boldsymbol{AA}^{\mathrm{T}}$ 的 n 个正交特征向量为 $\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n$. 不妨假设 $\|\boldsymbol{p}_i\|_2 = 1$，即有

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Ap}_i = \lambda_i \boldsymbol{p}_i .$$

对于任意 $\boldsymbol{x} \in \boldsymbol{R}^n$，令

$$\boldsymbol{x} = c_1\boldsymbol{p}_1 + c_2\boldsymbol{p}_2 + \cdots + c_n\boldsymbol{p}_n$$

$$\|\boldsymbol{x}\|_2^2 = c_1^2 + c_2^2 + \cdots + c_n^2$$

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Ax} = \lambda_1 c_1\boldsymbol{p}_1 + \lambda_2 c_2\boldsymbol{p}_2 + \cdots + \lambda_n c_n\boldsymbol{p}_n$$

$$\|\boldsymbol{Ax}\|_2^2 = (\boldsymbol{Ax})^{\mathrm{T}}(\boldsymbol{Ax}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Ax} = \boldsymbol{x}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{Ax})$$

$$= \lambda_1 c_1^2 + \lambda_2 c_2^2 + \cdots + \lambda_n c_n^2$$

$$\leqslant \lambda_n(c_1^2 + c_2^2 + \cdots + c_n^2)$$

$$= \lambda_n \|\boldsymbol{x}\|_2^2$$

$$\|\boldsymbol{A}\|_2 = \max_{\|\boldsymbol{x}\|_2 = 1}\|\boldsymbol{Ax}\|_2 \leqslant \sqrt{\lambda_n}$$

取 $\boldsymbol{x} = \boldsymbol{p}_n$ 可得

$$\|\boldsymbol{Ap}\|_2 = \sqrt{\lambda_n} .$$

【定理 5】 若 $\|\boldsymbol{B}\| < 1$，则 $\boldsymbol{I} \pm \boldsymbol{B}$ 非奇异，且

$$\left\|(\boldsymbol{I} \pm \boldsymbol{B})^{-1}\right\| \leqslant \frac{1}{1 - \|\boldsymbol{B}\|} .$$

【证明】用反证法证明 $\boldsymbol{I}\pm\boldsymbol{B}$ 非奇异.

若 $\boldsymbol{I}\pm\boldsymbol{B}$ 奇异，则存在 $\boldsymbol{x}\neq\boldsymbol{0}$，使得

$$(\boldsymbol{I}\pm\boldsymbol{B})\boldsymbol{x}=\boldsymbol{x}\pm\boldsymbol{B}\boldsymbol{x}=\boldsymbol{0},$$

$$\boldsymbol{x}=\mp\boldsymbol{B}\boldsymbol{x},$$

$$\|\boldsymbol{x}\|=\|\mp\boldsymbol{B}\boldsymbol{x}\|=\|\boldsymbol{B}\boldsymbol{x}\|\leqslant\|\boldsymbol{B}\|\|\boldsymbol{x}\|.$$

由此得 $\|\boldsymbol{B}\|\geqslant 1$，这与已知矛盾，所以 $\boldsymbol{I}\pm\boldsymbol{B}$ 非奇异.

现在证 $\left\|(\boldsymbol{I}\pm\boldsymbol{B})^{-1}\right\|\leqslant\dfrac{1}{1-\|\boldsymbol{B}\|}$，由

$$(\boldsymbol{I}+\boldsymbol{B})^{-1}(\boldsymbol{I}+\boldsymbol{B})=\boldsymbol{I}$$

得

$$(\boldsymbol{I}+\boldsymbol{B})^{-1}=\boldsymbol{I}-(\boldsymbol{I}+\boldsymbol{B})^{-1}\boldsymbol{B},$$

$$\|\boldsymbol{I}+\boldsymbol{B}\|^{-1}\leqslant\|\boldsymbol{I}\|+\left\|(\boldsymbol{I}+\boldsymbol{B})^{-1}\boldsymbol{B}\right\|\leqslant 1+\left\|(\boldsymbol{I}+\boldsymbol{B})^{-1}\right\|\|\boldsymbol{B}\|,$$

$$\left\|(\boldsymbol{I}+\boldsymbol{B})^{-1}\right\|(1-\|\boldsymbol{B}\|)\leqslant 1,$$

$$\left\|(\boldsymbol{I}+\boldsymbol{B})^{-1}\right\|\leqslant\frac{1}{1-\|\boldsymbol{B}\|}.$$

同样可证明

$$\left\|(\boldsymbol{I}-\boldsymbol{B})^{-1}\right\|\leqslant\frac{1}{1-\|\boldsymbol{B}\|}.$$

【推论】若 $\boldsymbol{A}$ 是非奇异矩阵，且 $\left\|\boldsymbol{A}^{-1}\right\|\|\delta\boldsymbol{A}\|<1$，则 $\boldsymbol{A}+\delta\boldsymbol{A}$ 非奇异.

【证明】因为

$$\left\|\boldsymbol{A}^{-1}\delta\boldsymbol{A}\right\|\leqslant\left\|\boldsymbol{A}^{-1}\right\|\|\delta\boldsymbol{A}\|<1,$$

所以 $\boldsymbol{I}+\boldsymbol{A}^{-1}\delta\boldsymbol{A}$ 非奇异，而

$$\boldsymbol{A}+\delta\boldsymbol{A}=\boldsymbol{A}(\boldsymbol{I}+\boldsymbol{A}^{-1}\delta\boldsymbol{A}).$$

由 $\boldsymbol{A}$ 和 $\boldsymbol{I}+\boldsymbol{A}^{-1}\delta\boldsymbol{A}$ 非奇异得 $\boldsymbol{A}+\delta\boldsymbol{A}$ 非奇异.

推论表明 $\boldsymbol{A}$ 是非奇异矩阵时，$\delta\boldsymbol{A}$ 中的元素绝对值充分小时 $\boldsymbol{A}+\delta\boldsymbol{A}$ 非奇异．这一性质在 2.4 会用到.

【定义 2】设 $\boldsymbol{A}$ 是 n 阶方阵，$\lambda_i(i=1,2,\cdots,n)$ 是 $\boldsymbol{A}$ 的特征值，则称

$$\rho(\boldsymbol{A})=\max_i|\lambda_i|$$

为方阵 $\boldsymbol{A}$ 的谱半径.

下面是谱半径和范数之间的关系.

【定理 6】设 $\boldsymbol{A}$ 为 n 阶方阵，$\|\boldsymbol{A}\|$ 是任何由向量范数导出的矩阵范数，$\rho(\boldsymbol{A})$ 是谱半径，则

$$\rho(\boldsymbol{A})\leqslant\|\boldsymbol{A}\|,$$

特别地，当 $\boldsymbol{A}^{\mathrm{T}}=\boldsymbol{A}$ 时有

$$\rho(\boldsymbol{A})=\|\boldsymbol{A}\|_2 .$$

【证明】设 λ 是 $\boldsymbol{A}$ 的特征值，$\boldsymbol{x}$ 是对应的特征向量，即 $\boldsymbol{Ax}=\lambda\boldsymbol{x}$，则由 $\boldsymbol{x}\neq\boldsymbol{0}$ 得 $\|\boldsymbol{x}\|>0$，所以

$$|\lambda|\leqslant\|\boldsymbol{A}\| .$$

由 λ 的任意性有

$$\rho(\boldsymbol{A})\leqslant\|\boldsymbol{A}\| .$$

当 $\boldsymbol{A}^{\mathrm{T}}=\boldsymbol{A}$ 时，记 $|\lambda_n|=\max\limits_i|\lambda_i|$，$\lambda_n^2$ 是 $\boldsymbol{A}^2$ 的特征值，也是 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ 的特征值，由此有

$$\|\boldsymbol{A}\|_2=\sqrt{\lambda_n^2}=|\lambda_n|=\rho(\boldsymbol{A}) .$$

【定理 7】对于任意给出的 $\varepsilon>0$，必定有一种由向量范数导出的矩阵范数 $\|\boldsymbol{A}\|_*$，使得

$$\|\boldsymbol{A}\|_*\leqslant\rho(\boldsymbol{A})+\varepsilon .$$

证明略.

2.4　直接法的稳定性分析

在没有测试误差和舍入误差的前提下，若满足书中所给的求解条件，用直接法所得的解都是理论解，因上述两种误差的存在，对于某些问题用上述所有算法都可获得满意解，对另一些问题有的能获得满意解，有的不能获得满意解，而对某些特殊问题，上述所有算法都难以获得满意解，只能用正交变换法求其近似解．本书不介绍正交变换法（第 5 章要介绍一种正交变换法，但不是解方程组）．具体要使用何种算法须对问题和算法做稳定性分析，而不是仅对问题做稳定性分析.

2.4.1　常见稳定性分析

先介绍一般教材中所做的稳定性分析.

1．右端项有一增量 $\delta\boldsymbol{b}$ 时对解的影响

设当 $\boldsymbol{b}$ 变为 $\boldsymbol{b}+\delta\boldsymbol{b}$ 时解变成 $\boldsymbol{x}+\delta\boldsymbol{x}$，即

$$\boldsymbol{A}(\boldsymbol{x}+\delta\boldsymbol{x})=\boldsymbol{b}+\delta\boldsymbol{b}$$

由 $\boldsymbol{Ax}=\boldsymbol{b}$ 得

$$\boldsymbol{A}\delta\boldsymbol{x}=\delta\boldsymbol{b}$$
$$\boldsymbol{x}=\boldsymbol{A}^{-1}\boldsymbol{b}$$
$$\delta\boldsymbol{x}=\boldsymbol{A}^{-1}\delta\boldsymbol{b}$$

两边取范数得

$$\|\delta\boldsymbol{x}\|\leqslant\|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{b}\|$$

$$\|\boldsymbol{b}\| = \|\boldsymbol{A}\boldsymbol{x}\| \leqslant \|\boldsymbol{A}\|\|\boldsymbol{x}\|$$

由以上两式得

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{\|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{b}\|}{\|\boldsymbol{x}\|} = \frac{\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{b}\|}{\|\boldsymbol{A}\|\|\boldsymbol{x}\|} \leqslant \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|} \tag{2.4-1}$$

式（2.4-1）表明，当右端项有一增量 $\delta\boldsymbol{b}$ 时，解的相对误差不超过右端项的相对误差和 $|\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$ 的乘积.

2．系数矩阵有一增量δA 时对解的影响

设此时解为 $\boldsymbol{x}+\delta\boldsymbol{x}$，即

$$(\boldsymbol{A}+\delta\boldsymbol{A})(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{b}$$

由 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ 得

$$\boldsymbol{A}\delta\boldsymbol{x} + \delta\boldsymbol{A}(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{0}$$

$$\delta\boldsymbol{x} = -\boldsymbol{A}^{-1}\delta\boldsymbol{A}(\boldsymbol{x}+\delta\boldsymbol{x})$$

当 $\|\delta\boldsymbol{A}\|$ 充分小时，即 $1 > 1 - \|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{A}\| > 0$ 时，有

$$\|\delta\boldsymbol{x}\| = \|\boldsymbol{A}^{-1}\delta\boldsymbol{A}(\boldsymbol{x}+\delta\boldsymbol{x})\| \leqslant \|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{A}\|(\|\boldsymbol{x}\| + \|\delta\boldsymbol{x}\|)$$

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leqslant \frac{\|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{A}\|}{1 - \|\boldsymbol{A}^{-1}\|\|\delta\boldsymbol{A}\|} = \frac{\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\dfrac{\|\delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}}{1 - \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\dfrac{\|\delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}} \tag{2.4-2}$$

通常将 $|\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$ 记为 $\mathrm{cond}_p(\boldsymbol{A})$，并称其为条件数，$p$ 可为 1, 2 和 ∞ 之一，当 $\mathrm{cond}_p(\boldsymbol{A})$ 相当大时，方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ 为病态方程.

实际上所有解方程组的数值解法都没有用 $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ 的．式（2.4-1）与式（2.4-2）和算法无关，最多只是对问题做稳定性分析，方程组的稳定性是和算法有关的.

【例 1】 分别用 Gauss 消元法、Gauss 列主元素法和改进平方根法解方程组

$$\begin{cases} 10^{-12}x_1 + 2x_2 = 4 + 2\times 10^{-12} \\ 2x_1 + x_2 = 22 \end{cases}$$

【解】 用单精度数解上面的方程组时，方程组变成

$$\begin{cases} 10^{-12}x_1 + 2x_2 = 4 \\ 2x_1 + x_2 = 22 \end{cases}$$

（1）用 Gauss 消元法的求解过程和结果为

$$\begin{cases} x_1 + 2\times 10^{12}x_2 = 4\times 10^{12} \\ -4\times 10^{12}x_2 = -8\times 10^{12} \end{cases}$$

解之得

$$x_2 = 2,\quad x_1 = 0$$

此时

$$\boldsymbol{U}_1 = \begin{bmatrix} 1 & 2\times 10^{12} \\ & 1 \end{bmatrix},\qquad \boldsymbol{U}_1^{-1} = \begin{bmatrix} 1 & -2\times 10^{12} \\ & 1 \end{bmatrix}$$

（2）用 Gauss 列主元素法的求解过程和结果为

$$\begin{cases} 10^{-12}x_1 + 2x_2 = 4 \\ 2x_1 + x_2 = 22 \end{cases}$$

$$\begin{cases} x_1 + 0.5x_2 = 11 \\ 2x_2 = 4 \end{cases}$$

解之得

$$x_2 = 2,\quad x_1 = 10$$

（3）用改进平方根法的计算过程和结果如下.

用单精度数将矩阵做三角分解得

$$\boldsymbol{A} = \begin{bmatrix} 10^{-12} & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \\ 0.5\times 10^{-12} & 1 \end{bmatrix} \begin{bmatrix} 10^{-12} & \\ & -0.25\times 10^{12} \end{bmatrix} \begin{bmatrix} 1 & 0.5\times 10^{12} \\ & 1 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} 1 & \\ 0.5\times 10^{12} & 1 \end{bmatrix},\ \boldsymbol{D} = \begin{bmatrix} 10^{-12} & \\ & -0.25\times 10^{12} \end{bmatrix},\ \boldsymbol{L}^{\mathrm{T}} = \begin{bmatrix} 1 & 0.5\times 10^{12} \\ & 1 \end{bmatrix}$$

令 $\boldsymbol{LDL}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{Ly} = \boldsymbol{b}, \boldsymbol{DL}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{y}$，有

$$\begin{bmatrix} 1 & \\ 0.5\times 10^{12} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 22 \end{bmatrix}$$

解之得

$$y_1 = 4,\quad y_2 = -2\times 10^{12}$$

此时 b_2 不起作用.

再令 $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{z}$，$\boldsymbol{D}\mathrm{z} = \boldsymbol{y}$，解之得

$$z_1 = 4\times 10^{12},\quad z_2 = 8$$

解 $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{z}$，得

$$x_2 = 8,\quad x_1 = 0$$

上面的方程组中，系数矩阵的逆是

$$\boldsymbol{A}^{-1} = \begin{bmatrix} -\dfrac{1}{4} & \dfrac{1}{2} \\ \dfrac{1}{2} & 0 \end{bmatrix}$$

$$\mathrm{cond}_1(\boldsymbol{A}) = 3\times\frac{3}{4} = \frac{9}{4}$$

按 2.4.1 节的结论，算法是稳定的．Gauss 消元法和改进平方根法所得解都应有意义，实际上只有 Gauss 列主元素所求解有意义，因此用条件数决定算法稳定性结论有问题．

2.4.2 消元法稳定性分析

【定理】对于方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，设右端项 $\boldsymbol{b}$ 有小扰动$\delta\boldsymbol{b}$，且 $\max|\delta b_i|=10^{-k}\varepsilon$．若用 Gauss 全主元素消元法解方程组．当

$$\begin{cases}\left\|a_i^{(i)}-a_{ji}^{(i-1)}a_j\right\|_\infty=M_{ij} & i=1,2,\cdots,n-1\\ a_i^{(i)}=(0,\cdots,0,1,a_{ii+1}^{(i)},\cdots,a_{in}^{(i)})^{\mathrm{T}}\\ a_j^{(i-1)}=(0,\cdots,0,a_{ji}^{(i-1)},\cdots,a_{jn}^{(i-1)})^{\mathrm{T}} & j=i+1,\cdots,n\end{cases}$$

上式中仅有

$$\begin{cases}M_{n-1n}=\varepsilon\\ M_{ij}=C(\text{常量}) & i=1,2,\cdots,n-2.\end{cases}$$

则

$$|\delta x_i|=O(10^{-k})\qquad i=1,2,\cdots,n.$$

【证明】设由$\delta\boldsymbol{b}$ 引起的解的变化为$\delta\boldsymbol{x}$，即

$$\boldsymbol{A}(\boldsymbol{x}+\delta\boldsymbol{x})=\boldsymbol{b}+\delta\boldsymbol{b}$$

由 $\boldsymbol{Ax}=\boldsymbol{b}$ 有

$$\boldsymbol{A}\delta\boldsymbol{x}=\delta\boldsymbol{b}\,.$$

Gauss 全主元素消元法消元过程为

$$\begin{cases}\delta b_i^{(i)}=\delta b_i^{(i-1)}/a_{ii}^{i-1} & i=1,2,\cdots,n-1\\ \delta b_j^{(i)}=\delta b_j^{(i-1)}-\dfrac{a_{ji}^{(i-1)}}{a_{ii}^{(i-1)}}\delta b_i^{(i-1)} & j=i+1,i+2,\cdots,n\end{cases}$$

对于 Gauss 全主元素法，$|a_{ji}^{(i-1)}/a_{ii}^{(i-1)}|\leqslant 1$，由已知有 $\delta b_i^{(i)}$ 均为小量．

由 Guss 全主元素消元法求解及回代算式有

$$\begin{aligned}|\delta x_n|&=|\delta b_n|/|a_{nn}^{(n-1)}|\\ &=|\delta b_n|/M_{n-1n}\\ &=O(10^{-k})\end{aligned}$$

$$|\delta x_i|=|\delta b_i^{(i)}-\sum_{j=i+1}^{n}a_{ij}^{(i)}\delta x_j|$$

由于$|a_{ij}|\leqslant 1$，所以可认为

$$|\delta x_i|=O(10^{-k}).$$

这样定理就得到证明．

对于 Gauss 列主元素消元法，消元的稳定性一样，但求解的回代中，由于$|a_{ij}^{(i)}|\leqslant 1$可能不成立，因而回代稳定性稍差．对于 Gauss 消元法，$M_{n-1n}=\varepsilon$，M_{ij} 为常量不成立．$|a_{ii}^{(i-1)}|$ 可

能相当小甚至为 0，所以消元和求解的稳定性都不保证.

若方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 用 Gauss 全主元素消元法求解时 $M_{n-2n-1}=\varepsilon_1$，$M_{n-1n}=\varepsilon_2$．ε_1 和 ε_2 都是小量时，消元的稳定性就无法保证，δx_i 的量级为 $\max \delta b_i/\varepsilon_1\varepsilon_2$．这样的方程组肯定是病态方程组．用正交变换法求解都困难．这可能是设计错误或数学模型有问题.

【例 2】解方程组

$$\begin{cases} 1.0001x_1+ & x_2+x_3=7.00009 \\ x_1+1.0001x_2+ & x_3=7.00018 \\ x_1+ & x_2+x_3=8 \end{cases}$$

上述方程组右端项的理论值为 7.0001，7.0002 和 8，由于测试误差变成了 7.00009，7.00018 和 8.

现用 Gauss 列主元素法（若 $\boldsymbol{A}=\boldsymbol{A}^{\mathrm{T}}$，则列主元素法和全主元素法相同）求解，求解过程为

$$\begin{bmatrix} 1.0001 & 1 & 2 & 7.00009 \\ 1 & 1.0001 & 1 & 7.00018 \\ 2 & 1 & 1 & 8 \end{bmatrix} \to \begin{bmatrix} 2 & 1 & 1 & 8 \\ 1 & 1.0001 & 1 & 7.00018 \\ 1.0001 & 1 & 2 & 11.00009 \end{bmatrix}$$

$$\to \begin{bmatrix} 1 & 0.5 & 1 & 4 \\ 0 & -0.5001 & -0.5 & -3.00018 \\ 0 & -0.49995 & -0.49995 & -2.99969 \end{bmatrix} \to \begin{bmatrix} 1 & 0.5 & 1 & 4 \\ 0 & 1 & 0.99980004 & 5.999160168 \\ 0 & -0.49995 & -0.49995 & -2.99969 \end{bmatrix}$$

$$\to \begin{bmatrix} 1 & 0.5 & 1 & 4 \\ 0 & 1 & 0.99980004 & 5.999160168 \\ 0 & 0 & 0.000099970001 & 0.0040987401 \end{bmatrix}$$

由此有

$$x_3=4.09996968,\quad x_2=1.90000884,\quad x_1=1.00001074.$$

解方程组时所用 $\min\limits_i \left|a_{ii}^{(i-1)}\right|=0.00009997001$，$\max\left|\delta b_i\right|=0.00002$，所得结果有意义．若用 Gauss 消元法，则结果毫无意义．若右端项分别为 7，7 和 8，此时 $\max\left|\delta b_i\right|=0.0002$，解将没有意义.

本例所解方程组可认为是病态方程．病态方程可定义如下.

【定义】对于方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，若经初等变换后，系数矩阵中行量 $\left\|\boldsymbol{a}_i-\boldsymbol{a}_{i+1}\right\|\leqslant\varepsilon$，则称方程组为病态方程.

新定义比用条件数作为衡量病态方程的可操作性好.

当矩阵 $\boldsymbol{A}$ 有一增量时，采用 Gauss 全主元素消元法和 Gauss 列主元素消元法消元时，对 $a_{ii}^{(i-1)}$ 不会有多大变化，因此不影响算法的稳定性.

2.4.3　三角分解法稳定性分析

对于 $\boldsymbol{LU}$ 法，由式（2.2-3）有当 $\left|u_{kk}\right|<\varepsilon$ 时，算法是不稳定的．对于 $\boldsymbol{LL}^{\mathrm{T}}$ 法和 $\boldsymbol{LDL}^{\mathrm{T}}$ 法，

由式（2.2-7）和式（2.2-12）知，当$|l_{ii}|<\varepsilon$，$|d_i|<\varepsilon$时，算法是不稳定的．由于当$\boldsymbol{Ax=b}$确定后，所有三角分解法的系数分解都只能按确定的顺序进行，不像 Gauss 列主元素和 Gauss 全主元素法那样，可以通过行、列变换，选取绝对值较大的元素，因而当$\boldsymbol{A}$矩阵的性质不好时，算法是不稳定的．

【例 3】解方程组

$$\begin{cases}1.0001x_1+x_2=3.00009\\ x_1+1.0001x_2=3.00018\end{cases}$$

本例中$\boldsymbol{A}$是正定的且对角严格占优，读者可用平方根法和改进平方根法求解，求得的解都无意义．

最后需特别指出的是，书中各算法前面的所有定理都是在没有舍入误差和测试误差的前提下得出的结论，而实际计算时舍入误差和测试误差是不可避免的．

2.4.4 直接法稳定性分析结论

直接法的稳定性既取决于原始方程组$\boldsymbol{Ax=b}$的系数矩阵性质，又取决于算法，还取决于右端初始误差．若对$\boldsymbol{A}$经初等变换后有两行（向量）之差的范数小于ε，或者所在行对角线元素的绝对值小于ε，则方程组为病态方程组．为使解有意义，对于书中所介绍的算法，可采用高精度仪器测量右端项的值，使右端项最大测试误差的阶小于ε的阶，程序至少使用双精度数，这类题不宜用三角分解法．算法的稳定性和$\boldsymbol{A}$的条件数不直接相关，研究病态方程解法是一科研课题，这里不便详述．

习题 2

1．用 Gauss 消元法解下列方程组，并求其系数行列式的值：

（1）$\begin{cases}x_1+3x_2+3x_3=5\\ 2x_1+3x_2+5x_3=5\\ 3x_1+4x_2+7x_3=6\end{cases}$　　（2）$\begin{cases}x_1+2x_2+2x_3-3x_4=8\\ x_1+4x_2-2x_3+3x_4=6\\ 2x_1+2x_2+4x_4=2\\ 3x_1-x_3+2x_4=1\end{cases}$

2．已知线性方程组$\begin{cases}10^{-2}x_1+x_2=1\\ x_1+x_2=2\end{cases}$的理论解为$x_1=\dfrac{100}{99}$，$x_2=\dfrac{98}{99}$．试分别用两位浮点数按无交换的 Gauss 消元法和 Gauss 列主元素法求解．

3．用$\boldsymbol{LU}$分解法解方程组

$$\begin{pmatrix}1&3&2&3\\2&5&3&-2\\-2&-2&3&5\\1&2&1&-2\end{pmatrix}\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}0\\7\\-1\\4\end{pmatrix}.$$

4．用追赶法解方程组

$$\begin{pmatrix} -4 & 1 & 0 & 0 \\ 1 & -4 & 1 & 0 \\ 0 & 1 & -4 & 1 \\ 0 & 0 & 1 & -4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

5．设方阵

$$\boldsymbol{A} = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix},$$

判断 $\boldsymbol{A}$ 是否为正定方阵．如果是，试求其 $\boldsymbol{LL}^{\mathrm{T}}$ 分解．

6．在 $\boldsymbol{R}^2$ 中画出满足下列条件的点集：

$$\|\boldsymbol{x}\|_1 \leqslant 1,\quad \|\boldsymbol{x}\|_2 \leqslant 1,\quad \|\boldsymbol{x}\|_\infty \leqslant 1.$$

7．证明 $\|\boldsymbol{A}\|_\infty = \max\limits_i \sum\limits_{j=1}^{n} |a_{ij}|$．

8．设方阵

$$\boldsymbol{A} = \begin{bmatrix} 5 & 7 & 3 \\ 7 & 11 & 2 \\ 3 & 2 & 6 \end{bmatrix},\quad \boldsymbol{B} = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}.$$

在范数 $\|A\|_1$，$\|B\|_\infty$ 意义下，分别计算 $\mathrm{cond}(\boldsymbol{A})$ 和 $\mathrm{cond}(\boldsymbol{B})$．

9．方程组

$$\begin{pmatrix} 1.001 & 0.25 \\ 0.25 & 0.0625 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.501 \\ 0.375 \end{pmatrix}$$

有解 $(1,\ 2)^{\mathrm{T}}$，将右端项变化

$$\Delta \boldsymbol{b} = \begin{pmatrix} -0.001 \\ 0.005 \end{pmatrix},$$

求解变化后的方程组 $\boldsymbol{Ax} = \boldsymbol{b} + \Delta \boldsymbol{b}$．

10．设 $\boldsymbol{A}$ 为非奇异方阵，$\boldsymbol{B}$ 是任一奇异方阵，则

$$\frac{1}{\|\boldsymbol{A} \pm \boldsymbol{B}\|} \geqslant \|\boldsymbol{A}^{-1}\|.$$

11．若 $\|\boldsymbol{A}\| < 1$，则

$$\|\boldsymbol{I} - (\boldsymbol{I} - \boldsymbol{A})^{-1}\| \leqslant \frac{\|\boldsymbol{A}\|}{1 - \|\boldsymbol{A}\|}.$$

12．证明

$$\frac{\|\boldsymbol{B}^{-1} - \boldsymbol{A}^{-1}\|}{\|\boldsymbol{B}^{-1}\|} \leqslant \mathrm{cond}(\boldsymbol{A}) \frac{\|\boldsymbol{A} - \boldsymbol{B}\|}{\|\boldsymbol{A}\|}.$$

13．证明定理 4、定理 5 和定理 6．

第 3 章　解方程 $f(x)=0$ 的迭代法

五次以上的代数方程无法找到理论解，一般超越方程也难以找到理论解，大都只能获得近似解．也就是说无法用直接法得到一般方程 $f(x)=0$ 的解，只能用迭代法求其满意解．迭代法是需要用无穷多次运算才能得到理论解的算法，由于运算次数总是有限的，所以无法得到理论解．

本章将介绍迭代法的思想，介绍解方程 $f(x)=0$ 的常用迭代法，介绍这些算法的收敛条件，还要介绍分离根的简易方法．

本章所介绍的算法只能求单根，不能求重根，重根是无法分离的．

3.1　逐次迭代法

对于一般方程而言，逐次迭代法仅是理论算法，但用隐式算法解常微分方程时常用逐次迭代法．此时则为实用算法．

3.1.1　逐次迭代法

对于方程

$$f(x)=0\ ,$$

其迭代公式推导过程如下．

1．将方程做恒等变换或同解变换

对于 $f(x)=0$，先将其转换成

$$x=\varphi(x)\ .$$

显然，由 $f(x)=0$ 经恒等变换或同解变换转换成 $x=\varphi(x)$，其转换方法不是唯一的，即使对于最简单的线性方程 $ax+b=0$，也可做无穷多个变换：

$$x=(a+1)x+b\ ,$$

$$x=(x^2+ax+b)^{1/2}\ ,$$

$$x=((x+1)^2+ax+b)^{1/2}-1\ ,$$

$$\cdots$$

2．变数学模型 $x=\varphi(x)$ 为计算公式

$$\begin{cases} x_{n+1}=\varphi(x_n) \\ x_0=a \end{cases} \tag{3.1-1}$$

按式（3.1-1）计算，若有

$$\lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} \varphi(x_n) = \alpha \ ,$$

则 α 为方程 $f(x)=0$ 的根，由此知用式（3.1-1）解题时存在如下三个问题：

（1） x_0 取何值；

（2）选何种 $x_{n+1}=\varphi(x_n)$ 算法收敛；

（3）收敛速度.

3．逐次迭代法的收敛条件

压缩映像原理可解决这一问题.

【定义 1】当函数 $\varphi(x)$ 在区间 $[a,b]$ 上满足下列两个条件时，称 $\varphi(x)$ 是 $[a,b]$ 上的压缩映像，式中 L 为压缩系数，也称为 Lipshitz 系数.

（1）封闭性条件： $\forall x\in[a,b]$，有 $\varphi(x)\in[a,b]$；

（2）压缩性条件： $\forall x,y\in[a,b]$，存在 $0<L<1$ 满足 $|\varphi(x)-\varphi(y)|\leqslant L|x-y|$.

【定理 1】若 $\varphi(x)$ 为 $[a,b]$ 上的压缩映像，则 $\varphi(x)$ 必是 $[a,b]$ 上的连续函数.

证明略.

【定理 2】设 $\varphi(x)$ 是 $[a,b]$ 上的压缩映像，则对任意 $x_0\in[a,b]$，由逐次迭代公式 $x_{n+1}=\varphi(x_n)$ 产生的序列 $\{x_n\}$ 收敛于 $f(x)=0$ 的唯一根 x^*，且有误差估计公式

$$\begin{cases} |x_n-x^*| \leqslant \dfrac{L}{1-L}|x_n-x_{n-1}| \\ |x_n-x^*| \leqslant \dfrac{L^n}{1-L}|x_n-x_0| \end{cases} \quad n=1,2,\cdots \tag{3.1-2}$$

【证明】由封闭性条件有 $a-\varphi(a)\leqslant 0$，$b-\varphi(b)\geqslant 0$，由定理 1 知 $\varphi(x_n)$ 是 $[a,b]$ 上的连续函数，方程 $x=\varphi(x)$ 在区间 $[a,b]$ 上有根 x^*.

设方程 $x=\varphi(x)$ 有另一根 $\tilde{x}\neq x^*$，则由

$$|x^*-\tilde{x}| = |\varphi(x^*)-\varphi(\tilde{x})| \leqslant L|x^*-\tilde{x}| < |x^*-\tilde{x}|$$

导出矛盾，所以方程 $x=\varphi(x)$ 在 $[a,b]$ 上有唯一根 x^*.

对于 $\forall x_0\in[a,b]$，由

$$|x_n-x^*| = |\varphi(x_{n-1})-\varphi(x^*)| \leqslant L|x_{n-1}-x^*| \leqslant L^2|x_{n-2}-x^*| \leqslant \cdots \leqslant L^n|x_0-x^*| \ ,$$

因 $0<L<1$，所以

$$\lim_{n\to\infty}|x_n-x^*|=0 \ ,$$

即

$$\lim_{n\to\infty} x_n = x^* \ .$$

同样因 $0<L<1$，由

$$\left|x_n - x^*\right| = \left|\varphi(x_{n-1}) - \varphi(x^*)\right| \leqslant L\left|x_{n-1} - x^*\right| \leqslant L\left|x_n - x^*\right| + L\left|x_n - x_{n-1}\right|$$

得

$$\left|x_n - x^*\right| = \left|\varphi(x_{n-1}) - \varphi(x^*)\right| \leqslant \frac{L}{1-L}\left|x_n - x_{n-1}\right| .$$

又因为

$$\left|x_n - x_{n-1}\right| = \left|\varphi(x_{n-1}) - \varphi(x_{n-2})\right| \leqslant L\left|x_{n-1} - x_{n-2}\right|$$

得

$$\left|x_n - x^*\right| = \left|\varphi(x_{n-1}) - \varphi(x^*)\right| \leqslant \frac{L}{1-L}\left|x_n - x_{n-1}\right| \leqslant \frac{L^2}{1-L}\left|x_{n-1} - x_{n-2}\right| \leqslant \cdots \leqslant \frac{L^n}{1-L}\left|x_1 - x_0\right| .$$

【推论】将定义中的第二条件改为 $\forall x \in [a,b]$，有

$$\varphi'(x) \leqslant L < 1 ,$$

则定理 2 的结论仍然成立.

【证明】 $\forall x, y \in [a,b]$，由微分中值定理知

$$\varphi(x) - \varphi(y) = \varphi'(\xi)(x - y) ,\quad \xi \in [x, y) ,$$

于是

$$\left|\varphi(x) - \varphi(y)\right| = \left|\varphi'(\xi)\right|\left|x - y\right| \leqslant L\left|x - y\right| ,$$

即 $\varphi(x)$ 满足压缩映像原理，故定理 2 的结论成立.

3.1.2 收敛阶

【定义 2】设 α 是方程 $y = \varphi(x)$ 的根，当 x_0 充分接近 α 时，由公式 $x_{n+1} = \varphi(x_n)$ 得到的序列 $\{x_n\}$ 收敛于 α，若存在常数 $p \geqslant 1$ 和正数 c，使得

$$\lim_{n\to\infty} \frac{\left|x_{n+1} - \alpha\right|}{\left|x_n - \alpha\right|^p} = c ,$$

则称序列 $\{x_n\}$ 是 p 阶收敛的，或称 $\{x_n\}$ 的收敛阶为 p，且称 $\varphi(x)$ 是 p 阶迭代函数.

显然 p 的大小刻画了迭代序列 $\{x_n\}$ 的收敛速度，p 越大收敛越快，$p = 1$ 时序列线性收敛（$c < 1$）；$1 < p < 2$ 时序列称为超线性收敛；$p = 2$ 时序列平方收敛.

下面的定理给出了收敛阶数的判别方法.

【定理 3】设迭代函数 $\varphi(x)$ 满足下列条件：

（1）设 α 的某一邻域有 $p(p \geqslant 2)$ 阶连续导数；

（2）$\alpha = \varphi(\alpha)$；

（3）$\varphi(\alpha) = \varphi'(\alpha) = \cdots = \varphi^{(p-1)}(\alpha) = 0$，$\varphi^{(p)}(\alpha) \neq 0$.

则当初始近似值 x_0 充分接近 α 时，由 $x_{n+1} = \varphi(x_n)$ 产生的序列 $\{x_n\}$ 是 p 阶收敛的，且

$$\lim_{n\to\infty} \frac{x_{n+1} - \alpha}{(x_n - \alpha)^p} = \frac{\varphi^{(p)}(\alpha)}{p!} .$$

【证明】由 Taylor 展开式得

$$x_{n+1}=\varphi(x_n)=\varphi(\alpha)+\varphi'(\alpha)(x_n-\alpha)+\cdots+\frac{\varphi^{(p-1)}(\alpha)}{(p-1)!}+\frac{\varphi^{(p)}(\xi)}{p!}(x_n-\alpha)^p$$

$$=\alpha+\frac{\varphi^{(p)}(\xi_n)}{p!}(x_n-\alpha)^p.$$

其中 ξ_n 在 α 和 x_n 之间，于是

$$\lim_{n\to\infty}\frac{x_{n+1}-\alpha}{(x_n-\alpha)^p}=\frac{\varphi^{(p)}_{(\xi)}}{p!}\neq 0\ .$$

3.1.3　逐次迭代法的几何意义

从几何上看，逐次迭代法把方程 $f(x)=0$ 的求根问题转换成 $x=\varphi(x)$ 并变成求序列 $\{x_n\}$ 的极限．这实际上是把求根问题转化为求直线 $y=x$ 与曲线 $y=\varphi(x)$ 的交点 (α,α)，该交点的横坐标就是 $x=\varphi(x)$ 或 $f(x)=0$ 的根．迭代过程如图 3.1 所示．由 x_n 求 x_{n+1} 的过程为：先过点 (x_0,x_0) 作 x 轴的垂线交 $y=\varphi(x)$ 于点 $(x_0,\varphi(x_0))$；再过点 $(x_0,\varphi(x_0))$ 作 y 轴的垂线交直线 $y=x$ 于点 (x_1,x_1) 求得点 (x_n,x_n) 之后，过点 (x_n,x_n) 作 x 轴的垂线，交曲线 $y=\varphi(x)$ 于点 $(x_n,\varphi(x_n))$，即 (x_n,x_{n+1})．

由图 3.1（a）和图 3.1（b）可看出，序列 $\{x_n\}$ 愈来愈靠近根 α，由图 3.1（c）和图 3.1（d）可看出序列 $\{x_n\}$ 越来越远离曲线 $y=\varphi(x)$ 和直线 $y=x$ 的交点；前者收敛，后者发散．

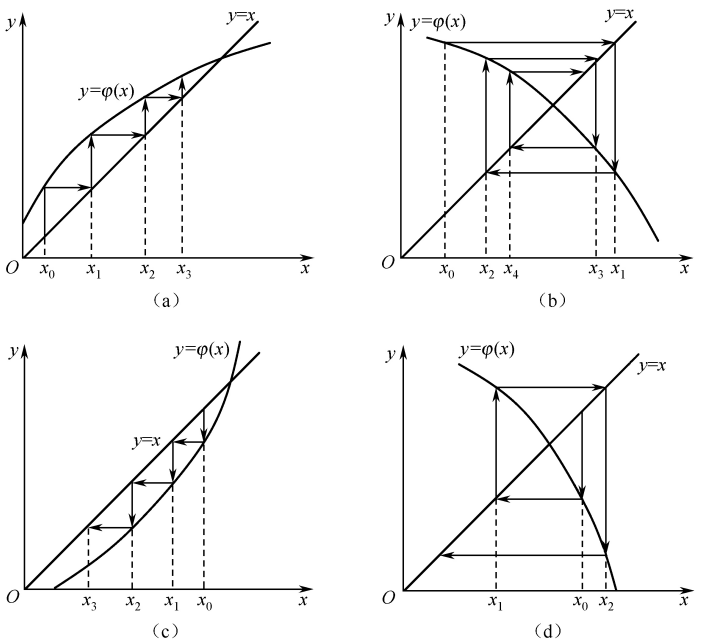

图 3.1　逐次迭代法的迭代过程

可以通过作图法确定根的近似值和压缩映像区间.

3.1.4 计算实例

【例 1】 用逐次迭代法求方程

$$x\mathrm{e}^{x}-\frac{1}{4}=0$$

的实根（准确到10^{-3}）.

【解】 $f(x)=x\mathrm{e}^{x}-\frac{1}{4}$是一个单调上升函数，由于$f(0)=-\frac{1}{4}<0$，$f(1)=\mathrm{e}-\frac{1}{4}>0$，所以根$x^{*}\in[0,1]$.

对$f(x)=0$做恒等变换，取

$$x=\frac{1}{4\mathrm{e}^{x}}=\varphi(x),\quad \varphi(0)\in[0,1],\quad 0<\varphi(0)=\frac{1}{4\mathrm{e}}<\frac{1}{4}<1.$$

当$x\in[0,1]$时，$0<|\varphi'(x)|<\frac{1}{4}$，逐次迭代法收敛.

取$x_0=0$，计算公式为

$$\begin{cases} x_{n+1}=\varphi(x_n)=\dfrac{1}{4\mathrm{e}^{x_n}}, & n=0,1,\cdots \\ x_0=0, & |x_{n+1}-x_n|>\varepsilon=10^{-3} \end{cases}$$

算式中$|x_{n+1}-x_n|>\varepsilon$表示迭代条件．一旦$|x_{n+1}-x_n|\leqslant\varepsilon$（迭代精度），则迭代结束．计算结果见表 3.1.

表 3.1 逐次迭代法计算$x_{n+1}=\frac{1}{4\mathrm{e}^{x_n}}$的计算结果

n	x_n	$f(x_n)$
0	0	
1	0.250000	−0.250000
2	0.194700	0.071007
3	0.205770	0.013450
4	0.203505	−0.000566
5	0.203967	0.000115

【例 2】 用逐次迭代法求方程$f(x)=x^3+4x^2-10=0$在区间$[1,2]$内的根．要求迭代精度为10^{-6}.

【解】 做恒等变换或同解变换：

（1）$x=\varphi_1(x)=x-x^3-4x^2+10$；

（2）$x=\varphi_2(x)=\left(\dfrac{10}{4+x}\right)^{1/2}$；

（3）$x=\varphi_3(x)=x-\dfrac{x^3+4x^2-10}{3x^2+8x}$.

$\varphi'(x)=1-3x^2-8x$．当 $x>1$ 时，$|\varphi'(x)|>1$，所以用 $x-x^3-4x^2+10$ 作为迭代函数，不收敛．而对于 $\varphi_2(x)$，当 $x\in[1,2]$ 时，$\varphi_2(x)\in[1,2]$.

$\varphi_2'(x)=-\dfrac{1}{2}\dfrac{\sqrt{10}}{\sqrt{(4+x)^3}}$．当 $x=1.5$ 时，$\varphi_2'(x)<1$，所以可取 $x_0=1.5$，用 $\varphi_2(x)$ 作为逐次迭代计算.

同样可知 $\varphi_3'(1.5)<1$，所以也可用 $\varphi_3(x)$ 作逐次迭代计算.

用 $\varphi_1(x)$，$\varphi_2(x)$，$\varphi_3(x)$ 做迭代函数的计算结果见表 3.2.

表 3.2　例 2 的计算结果表

n	$x_{n+1}=x_n-x_n^3-4x_n^2+10$	$x_{n+1}=\left(\dfrac{10}{4+x_n}\right)^{1/2}$	$x_{n+1}=x_n-\dfrac{-x_n^3-4x_n^2+10}{3x_n^2+8x_n}$
0	1.500000	1.500000000	1.500000000
1	−0.875000	1.348399725	1.373333333
2	6.732000	1.367376372	1.365262015
3	−469.700000	1.364957015	1.365230014
4	1.03×10^8	1.365235594	1.365230014
5		1.365230576	
6		1.365230023	
7		1.365230012	
8		1.365230014	

$\varphi_3(x)$ 用了同解变换，变换过程为 $f(x)=x^3+4x^2-10=0$，它和 $\dfrac{x^3+4x^2-10}{3x^2+8x}$ 同解，方程两端同加 $-x$ 也同解．将 $f(x)$ 除以在区间 $[1,2]$ 上不为 0 的 $3x^2+8x$ 的目的在于，保证迭代函数的封闭性和压缩性.

用逐次迭代法解方程 $f(x)=0$ 做恒等变换或同解变换时，变换是一对多的，而且有不少变换不能用做迭代函数，因而一般情况下实用性不强．但是若方程本身的表述形式就是 $x=\varphi(x)$ 且收敛性又能保证，则使用起来很方便（常微分方程数值算法中所有隐式格式正好满足上述条件）．另外这种思想或解题途径可借用于解方程组.

3.2　Newton 法

用逐次迭代法解方程时，将 $f(x)=0$ 转换成 $x=\varphi(x)$ 不是唯一的，而且不能给出通用转换方法，原因是 $f(x)$ 所表述的函数太多太广．Newton 法采用的方法是将 $f(x)$ 用它的一阶

Taylor 级数代替，或者用切线代替曲线 $y=f(x)$ 以确保由 $f(x)=0$ 转换成 $x=\varphi(x)$ 的唯一性（收敛性并不能保证）.

3.2.1 Newton 法算式推导

对于函数 $y=f(x)$，其一阶导数连续，过点 (x_n,y_n) 的切线方程为

$$y=y_n+f'(x_n)(x-x_n)\,. \tag{3.2-1}$$

式（3.2-1）也正是 $f(x)$ 在 x_n 点展开的一阶 Taylor 级数．令切线的根的 x 坐标为 x_{n+1}，则

$$x_{n+1}=x_n-\frac{y_n}{f'(x_n)}=x_n-\frac{f(x_n)}{f'(x_n)},\quad |x_{n+1}-x_n|>\varepsilon,\quad n=0,1,\cdots \tag{3.2-2}$$

当 $|x_{n+1}-x_n|\leqslant\varepsilon$ 时式（3.2-2）结束.

式（3.2-2）称为解方程 $f(x)=0$ 的 Newton 法.

3.2.2 Newton 法的几何意义

Newton 法的求根思想就是不断用切线的根代替曲线的根，直到得到满意根．设 x_n 是 $f(x)=0$ 的一个近似根，过点 $(x_n,f(x_n))$ 作切线

$$y=f(x_n)+f'(x_n)(x-x_n)\,.$$

求该切线的根，即求

$$0=f(x_n)+f'(x_n)(x-x_n)$$

的根 x，并令其为 x_{n+1}，由上式知

$$x_{n+1}=x_n+\frac{f(x_n)}{f'(x_n)},\quad n=0,1,\cdots$$

这正是式（3.2-2），不断做这一工作，直到 $|x_{n+1}-x_n|\leqslant\varepsilon$ 为止，其几何图形如图 3.2 所示.

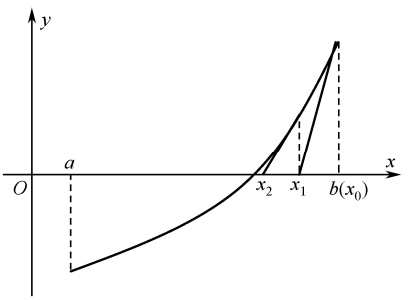

图 3.2 Newton 法的几何背景

正因为 Newton 法有上述几何意义，故 Newton 法也称为切线法.

3.2.3 Newton 法的收敛条件

【定理 1】设函数 $f(x)$ 在区间 $[a,b]$ 上存在二阶导数，且满足条件：

（1） $f(a)f(b)<0$；

（2） $f'(x)\neq 0$， $x\in[a,b]$；

（3） $f''(x)$ 在 $[a,b]$ 上不变号；

（4）取初始值 $x_0\in[a,b]$，使得 $f(x_0)f''(x_0)>0$.

则由 Newton 迭代公式产生的序列 $\{x_n\}$ 收敛于方程 $f(x)=0$ 在 $[a,b]$ 上的唯一根 α.

【证明】条件（1）表明方程 $f(x)=0$ 在 $[a,b]$ 上有根．条件（2）保证了式（3.2-2）能顺利运行．条件（3）表明 $f(x)=0$ 在区间 $[a,b]$ 上有唯一根.

满足条件（4）的 $f(x)$ 有四种情况，如图 3.3 所示.

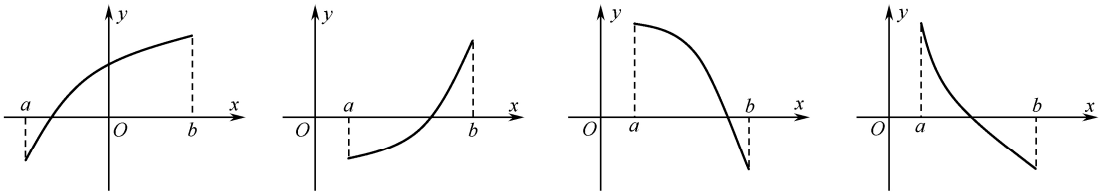

图 3.3　Newton 法收敛定理 1 的几何背景

现在证明其中一种情况：

$$f(a)<0,\quad f(b)>0,\quad f'(x)>0,\quad f''(x_0)\geqslant 0.$$

根据条件（4）有 $f(x_0)>0$，下面证明序列 $\{x_n\}$ 严格单调下降.

$$x_1=x_0-\frac{f(x_0)}{f'(x_0)}<x_0.$$

另一方面，$f(x)$ 在其根 x^* 处的 Taylor 展开式为

$$f(x^*)=f(x_0)+f'(x_0)(x^*-x_0)+\frac{1}{2}f''(\xi_0)(x^*-x_0)^2=0.$$

ξ_0 在 x_0 和 x^* 之间. 所以有

$$f(x_0)+f'(x_0)(x^*-x_0)=-\frac{1}{2}f''(\xi_0)(x^*-x_0)^2,$$

$$x^*-\left(x_0-\frac{f(x_0)}{f'(x_0)}\right)=-\frac{1}{2}\frac{f''(\xi_0)}{f'(x_0)}(x^*-x_0)^2.$$

即

$$x^*-x_1=-\frac{1}{2}\frac{f''(\xi_0)}{f'(x_0)}(x^*-x_0)^2.$$

所以

$$x^*<x_1<x_0.$$

因为 $x_1>x^*$，由 $f(x)$ 的单调性知 $f(x_1)>f(x^*)=0$.

由 Newton 迭代公式有

$$x_2=x_1-\frac{f(x_1)}{f'(x_1)}.$$

因为 $f(x_1)>0$, $f'(x_1)>0$，所以 $x_2<x_1$.

一般地，由 $x^*\leqslant x_n\leqslant x_0$ 及 $f(x)>0$ 可得 $f(x_n)>f(x^*)=0$，因此

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}<x_n,$$

$$x^* - x_{n+1} = -\frac{1}{2}\frac{f''(\xi_n)}{f'(\xi_n)}(x^* - x_0)^2 \leqslant 0,$$

$$x^* \leqslant x_{n+1} < x_0, \quad n = 0,1,\cdots$$

由此得 $\{x_n\}$ 为单调下降序列，且以 $f(x)=0$ 的根 x^* 为下限.

因 $\{x_n\}$ 为单调下降序列且有下限，所以有极限. 设极限为 x，对式（3.2-2）两端取极限得

$$x = x - \frac{f(x)}{f'(x)},$$

从而有 $f(x)=0$.

由于极限是唯一的，所以有

$$x = x^*.$$

同样可证明另外三种情况.

【定理 2】设 $f(x)=0$ 的根 α 的邻域有二阶连续导数，$f'(\alpha)\neq 0$，则当 x_0 充分接近于 α，即在 α 的邻域内时，由 Newton 法所产生的序列 $\{x_n\}$ 收敛于方程 $f(x)=0$ 的根 α，且收敛阶为 2.

【证明】由 Newton 法迭代公式有

$$\varphi(x) = x - \frac{f(x)}{f'(x)},$$

$$\varphi'(x) = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = -\frac{f(x)f''(x)}{(f'(x))^2}.$$

按 3.1 节的定理 3 有 Newton 法收敛且收敛阶为 2.

3.2.4 Newton 法的计算过程和计算实例

Newton 法解 $f(x)=0$ 的计算过程如下.

（1）写出迭代算式，给出迭代精度 ε.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

（2）确定 x_0. 对于 $f''(x)$ 在有根区间不变号的函数，按 3.2 节的定理 1 选择 x_0；对于变化较复杂的函数，则需要分离根，按 3.2 节的定理 2 选取 x_0.

（3）按迭代公式求根.

【例 1】用切线法求 $c^{1/p}$，其中 c 为正数，p 为整数，当 $p=3$，$c=441$ 时，计算 $\sqrt[3]{441}$，$\varepsilon = 10^{-5}$.

【解】令 $f(x)=x^p - c=0$，则问题变成求方程 $f(x)=x^p-c$ 的根. 因 $f'(x)=px^{p-1}$，所以 Newton 法迭代公式为

$$x_{n+1}=x_n-\frac{x_p^n-c}{px_n^{p-1}}=\begin{cases}\dfrac{1}{p}((p-1)x_n+c/x_n^{p-1}), & p>0\\ -\dfrac{x_n}{p}((1-p)-cx_n^{-p}), & p<0\end{cases}$$

特别地，当 $p=3$ 时，上式为求 c 的立方根，显然，$f'(x)>0$，$f''(x)>0$.

适当地选取初始近似值 x_0，只须迭代很少几次就能得到足够精确的解.

当 $p=3$，$c=441$ 时，迭代公式为

$$x_{n+1}=\frac{1}{3}\left(2x_n-\frac{441}{x_n^2}\right).$$

因 $f(7)<0$，$f(8)>0$，所以 $x^*\in[7,8]$，取 $x_0=8$［与 $f''(x)$ 同号］，$\varepsilon=10^{-5}$，则迭代结果如表 3.3 所示。

表 3.3　Newton 法计算 $\sqrt[3]{441}$ 的结果

n	x_n	$f(x_n)$
1	8.000000	71.0000000
2	7.630203	3.2313000
3	7.6117077	0.0078000
4	7.6116626	0.0000019
5	7.6116626	0.0000019

由上表知 $\sqrt[3]{441}=7.6116626$.

牛顿法算式简单，收敛阶高，但每次运算时要计算系数的导数值，为了避免导数计算，人们给出了割线法.

3.3　割线法

为了避免导数计算，可用割线代替切线. 割线法就是用割线的根代替切线的根的解方程算法. 割线方程须过两点，当一点固定不变时，称为单点割线法，当两点都变时称为双点割线法.

3.3.1　单点割线法

用单点割线法求 $f(x)=0$ 的迭代公式的几何背景如图 3.4 所示.

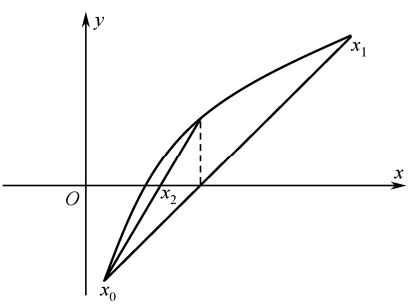

图 3.4　单点割线法的几何背景

过点 x_0 和点 x_n 的直线公式为

$$\frac{y-f(x_n)}{x-x_n}=\frac{y-f(x_0)}{x-x_0}\qquad(3.3\text{-}1)$$

令 $y=0$ 时，割线与 x 轴的交点的横坐标为 x_{n+1}，则

$$x_{n+1}=x_n-\frac{x_n-x_0}{f(x_n)-f(x_0)}f(x_n)，\quad n=1,2,\cdots \tag{3.3-2}$$

实际上，若直线用割线的斜率代替切线的斜率，可直接得到单点割线法的计算公式：

$$x_{n+1}=x_n-\frac{f(x_n)}{\dfrac{f(x_n)-f(x_0)}{x_n-x_0}}，\quad n=2,3,\cdots，\quad |x_{n+1}-x_n|>\varepsilon \tag{3.3-3}$$

式（3.3-3）将式（3.3-2）表述得更完善，加上了迭代条件 $|x_{n+1}-x_n|>\varepsilon$．从便于记忆的角度讲，式（3.3-3）比式（3.3-2）好一些.

3.3.2 单点割线法的收敛条件

从图 3.1（c）可以看出，当 x_0 和 x_1 的序号互异时，迭代不收敛.

【定理 1】对于 $f(x)$，$x\in[a,b]$，满足以下三个条件：

（1）$f(a)f(b)<0$；

（2）$f''(x)$ 不变号；

（3）$f''(x)f(x_0)>0$，$f''(x)f(a)>0$ 取 $x_0=a$，否则取 $x_0=b$.

这一定理的证明和 3.2 节中定理的证明类似，证明略.

【定理 2】在 $f(x)=0$ 的根的邻域 I 内任取 $x_0,x_1\in I$，用单点割线法求根收敛.

证明略.

3.3.3 单点割线法的计算过程和计算实例

单点割线法的计算过程如下.

（1）将 $f(x)=0$ 转换成迭代公式，给出迭代精度：

$$x_{n+1}=x_n-\frac{f(x_n)}{\dfrac{f(x_n)-f(x_0)}{x_n-x_0}}$$

（2）按 3.3 节的定理 1 或定理 2 选择适当的 x_0.

（3）利用迭代公式做迭代计算.

【例 1】用单点割线法求 $f(x)=x^3-2x-5$ 在 $[2,3]$ 内的根（迭代精度为 10^{-5}）.

【解】取 $x_0=3,x_1=2$，则

$$x_{n+1}=x_n-\frac{x_n^3-2x_n-5}{\dfrac{(x_n^3-2x_n-5)-(3^3-2\times3-5)}{x_n-3}}.$$

具体计算结果如表 3.4 所示.

表 3.4　单点割线法的计算结果

n	x_n	$f(x_n)$
0	3	16
1	2	−1
2	2.058823529	−0.390799923
3	2.096558636	0.022428052
4	2.094440519	−0.001238424
5	2.094557622	0.000068537
6	2.094551140	−0.000003812

由上表知 $f(x)$ 的根为 2.094551140.

3.3.4　双点割线法

对于单点割线法，由于 x_0 始终不变，故影响了收敛速度．单点割线法的收敛阶为 1，双点割线法的实质是用差商代替导数的牛顿法．迭代公式为

$$\begin{cases} x_{n+1}=x_n-\dfrac{f(x_n)}{\dfrac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}}, \quad n=2,3,\cdots, \quad |x_{n+1}-x_n|>\varepsilon \\ x_0, x_1 \text{已知} \end{cases} \tag{3.3-4}$$

双点割线法的收敛阶为 $\dfrac{\sqrt{5}-1}{2}\doteq 1.618$.

3.3.5　双点割线法的收敛条件

双点割线法的收敛条件的相关定理及其证明和单点割线法的类似，证明和叙述略.

3.3.6　双点割线法的计算过程和计算实例

双点割线法的计算过程如下.

（1）写出迭代公式，给出迭代精度：

$$x_{n+1}=x_n-\frac{f(x_n)}{\dfrac{f(x_n)-f(x_0)}{x_n-x_0}}$$

（2）按定理 1 决定区间的两个端点哪个作为 x_0，哪个作为 x_1 .

（3）按迭代公式求根.

【例 2】 用双点割线法求方程 $f(x)=x^3+4x^2-10=0$ 在区间 [1,1.5] 内的根（迭代精度为 2×10^{-3}).

【解】 取 $x_0=1$，$x_1=1.5$，用双点割线法公式：

$$\begin{cases} x_{n+1}=x_n-\dfrac{x_n^3+4x_n^2-10}{\dfrac{(x_n^3+4x_n^2-10)(x_{n-1}^3+4x_{n-1}^2-10)}{x_n-x_{n-1}}}, & n=2,3,\cdots \\ |x_{n+1}-x_n|>2\times10^{-3} \\ x_0=1 \\ x_1=1.5 \end{cases}$$

计算结果如表 3.5 所示.

表 3.5 双点割线法的计算结果

n	x_n	$f(x_n)$
0	1.0000000	−5.0000000
1	1.5000000	2.3750000
2	1.3389831	−0.4278675
3	1.3635629	−0.0275080
4	1.3652547	0.0003579

由上表知 $f(x)$ 的根为 1.3652547.

3.4 对分法

对分法是一种古老的算法，对分法求解方程的思想是不断地对分有根区间，使有根区间达到足够小，或者使方程中左端函数的绝对值足够小．对分法的计算过程和迭代法一样，需要计算无限次才能获得理论解.

3.4.1 对分法算式推导

对于方程 $f(x)=0$，$x\in[a,b]$，若 $f(a)f(b)<0$ 且在 $x\in[a,b]$ 上只有唯一根 α，则可用对分法求其根.

令 $a_0=a$，$b_0=b$，$c_0=\dfrac{a+b}{2}$，则对分法求有根区间左端点、右端点及中点的计算公式为

$$\begin{cases} \begin{cases} a_i=\begin{cases} c_{i-1} & f(a_{i-1})f(c_{i-1})>0 \\ a_{i-1} & f(a_{i-1})f(c_{i-1})<0 \end{cases} \\ b_i=\begin{cases} c_{i-1} & f(b_{i-1})f(c_{i-1})>0 \\ b_{i-1} & f(b_{i-1})f(c_{i-1})>0 \end{cases} \\ c_i=\dfrac{a_i+b_i}{2} \end{cases} & i=1,2,\cdots \\ \begin{cases} b_i-a_i>\varepsilon \text{ 或} \\ |f(c_i)|>\varepsilon \end{cases} \end{cases} \qquad (3.4\text{-}1)$$

对分法的几何背景如图 3.5 所示.

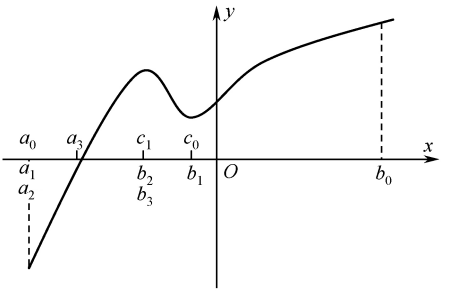

图 3.5　对分法的几何背景

与逐次迭代法、Newton 法和割线法不同，对分法所求得的 c_i 不能保证 i 愈大愈接近根. 因而在式（3.4-1）的迭代条件中还选取了 $|f(c_i)|>\varepsilon_i$. 实际上，前面所介绍的算法中也可用 $|f(x_n)|>\varepsilon$ 为另一迭代条件.

3.4.2　对分法的计算过程和计算实例

对分法的计算过程如下.

（1）给出计算公式和迭代精度；

（2）按式（3.4-1）求根.

【例 1】 用对分法计算 $x^3+10x-20=0$ 的唯一实根，$\varepsilon=10^{-4}$.

【解】 由于 $f(1)=-9, f(2)=8$，所以取 $a_0=-9$，$b_0=2$，用对分法求根，计算结果如表 3.6 所示.

表 3.6　对分法的计算结果

n	a_i	b_i	c_i	$f(x_i)$
0	1.0000000	2.0000000	1.5000000	−1.6250000
1	1.5000000	2.0000000	1.7500000	2.8593750
2	1.5000000	1.7500000	1.6250000	0.5410156
3	1.5000000	1.6250000	1.5625000	−0.5603027
4	1.5625000	1.6250000	1.5937500	−0.0143127
5	1.5937500	1.6250000	1.6093750	0.2621727
6	1.5937500	1.6093750	1.6015625	0.1236367
7	1.5937500	1.6015625	1.5976563	0.0545889
8	1.5937500	1.5976563	1.5957031	0.0201198
9	1.5937500	1.5947266	1.5947266	0.0028989
10	1.5937500	1.5947266	1.5942383	0.0059087
11	1.5942383	1.5947266	1.5944824	−0.0014048
12	1.5944824	1.5947266	1.5946045	−0.0017470
13	1.5944824	1.5946405	1.5945435	−0.0003289

由上表知解出的根为 1.5945435.

3.5　分离根方法及求所有根算法

逐次迭代法解方程时，由于 $f(x)=0$ 转换成 $x=\varphi(x)$ 没有规律可循，且收敛性不能保证，因而一般情况下，逐次迭代法不是实用算法. 使用 Newton 法、割线法解方程时，虽然计算公式是唯一的，但是用 Newton 或割线法求根时，要考虑收敛性. 对分法虽稳定性好，但收

敛慢，所有算法都只能求一个根，如果在 $x\in[a,b]$ 上有两个以上的根，所有算法都无能为力，因此必须分离根，即要求所求根区间内只有一个根. 对于 Newton 法和割线法，由于在 $f(x)=0$ 的根的邻域内 Newton 法和单、双点割线法都收敛，因而给分离根也提供了一个途径.

这里须附带提及的是，一个判别型定理的优劣主要取决于定理的可操作性，以 Newton 法为例，显然定理 2 的可操作性较好，因定理 1 中确定二阶导数单调太难，至少尚未见到一个有效的判别二阶导数单调的数值算法.

3.5.1 分离根方法

对于方程

$$f(x)=0,\quad x\in[a,b]$$

分离根的方法很简单，将区间 n 等分，当 n 充分大时，所有子区间 $[a+(i-1)h,a+ih]$，$h=(b-a)/n$ 都充分小，任何一个子区间至多只有一个根，这样就实现了根的分离.

显然，若 $f(a+(i-1)h)f(a+ih)\leqslant 0$，则该区间有一个根，否则无根.

分离根的结果不仅使每个子区间最多只有一个根，而且每个有根子区间的任一点都可认为是在根的邻域之中，这样就可使书中介绍的所有算法都收敛，都能求其满足迭代精度的根.

3.5.2 求所有根算法

求 $[a,b]$ 上所有根的计算过程如下（以 Newton 法为例）.

（1）将 $[a,b]$ 分成 n 等份：

$$\begin{cases} h=(b-a)/n \\ x_i=a+ih=x_{i-1}+h \end{cases} \qquad i=1,2,\cdots,n\ .$$

（2）用 Newton 法求根，计算公式为

$$\begin{cases} z_0=\dfrac{x_{i-1}+x_i}{2} \\ z_k=z_{k-1}-\dfrac{f(z_{k-1})}{f'(z_{k-1})} \qquad k=1,2,\cdots \\ f(x_{i-1})f(x_i)\leqslant 0 \\ \left|f(z_k)\right|>\varepsilon \end{cases} \tag{3.5-1}$$

式（3.5-1）中的 Newton 法公式也可用割线法公式、对分法公式代替.

3.5.3 特殊处理

用上述方法求根可能会出现增根，当根 $\alpha_j\approx x_i$ 时，在子区间 $[x_i,x_{i+1}]$ 上也可能满足 $f(x_i)f(x_{i+1})\leqslant 0$. 因为计算机中的 0 不是数学中的 0，为了防止增根，当在第 i 个子区间内找到一个根后，下一个判别有根无根的子区间不是第 $i+1$ 个子区间，而是第 $i+2$ 个子区间. 实际上，当 n 充分大后，两个子区间内最多只有一个根同样成立.

3.5.4　计算实例

【例 1】用 Newton 法求 $x^4-10x^3+35x^2-40x+24=0$ 的所有根 $(x\in[0,5])$，迭代精度 $\varepsilon=10^{-5}$.

【解】取 $n=100$，$h=0.005$，具体计算公式（在所有有根子区间）为

$$\begin{cases} z_0=\dfrac{x_{i-1}+x_i}{4} \\ z_k=z_{k-1}-\dfrac{z_{k-1}^4-10z_{k-1}^3+35z_{k-1}^2-50z_{k-1}+24}{4z_{k-1}^3-30z_{k-1}+70z_{k-1}-50} \\ \left|f(z_k)\right|>10^{-5} \end{cases}$$

计算结果为

$z_1=0.999989$，$z_2=1.999997$，$z_3=3.000003$，$z_4=4.000011$.

习题 3

1．用对分法求方程 $x^3-2x-5=0$ 在区间 $[2,3]$ 内的实根的近似值，要求精确到 $\dfrac{1}{2}\times10^{-3}$.

2．用 Newton 法求下列方程的根，要求具有四位有效数字：

（1）$x^3-x^2-x-1=0$ 的正根，$x_0=1.5$；

（2）$2x=\mathrm{e}^{-x}$, $x_0=0.3$；

（3）$\cos x=0.5+\sin x$ 的最小正根，$x_0=\dfrac{\pi}{8}$.

3．分别用单点割线法和双点割线法求方程 $x^3-3x-1=0$ 在区间 $[1,2]$ 内的根的近似值，要求 $\left|x_{n+1}-x_n\right|<10^{-3}$.

4．基于逐次迭代法原理证明

$$\sqrt{2+\sqrt{2+\sqrt{2+\cdots}}}=2.$$

5．求方程 $x^3-x^2-1=0$ 在 $x_0=1.5$ 附近的一个根，要求具有五位有效数字．若把方程写成下列不同的等价形式，讨论以上方法在区间 $[1.3,1.6]$ 上的敛散性，并选出收敛速度最快的迭代公式求根：

（1）$x=1+\dfrac{1}{x^2}$，相应的迭代公式为 $x_{n+1}=1+\dfrac{1}{x_n^2}$；

（2）$x^3=1+x^2$，相应的迭代公式为 $x_{n+1}=\sqrt[3]{1+x_n^2}$；

（3）$x^2=\dfrac{1}{x-1}$，相应的迭代公式为 $x_{n+1}=\dfrac{1}{\sqrt{x_n-1}}$.

用对分法求 $f(x)=x^3-3.6x^2+3.75x-1.1=0$ 在 $x\in[0,3]$ 上的所有根.

第 4 章　解线性代数方程组的迭代法

迭代法是解线性方程组的又一类算法．特别是对一些大型工程物理问题，至今仍然主要用迭代法求解，其原因是迭代法可只存储非零元素，而变带宽压缩存储平方根法和改进平方根法要存储带内所有元素．对于许多大型二维和三维力学问题，带内零元素相当多，只能用迭代法求解，对于阶数较高的线性方程组，（压缩存储）迭代法内存开销小，计算量小，且容易控制误差．

解线性方程组的迭代法的基本思想与解方程的逐步迭代法类似，即把性质良好的方程组

$$\boldsymbol{Ax} = \boldsymbol{b}$$

按一定途径转换成

$$\boldsymbol{x} = \boldsymbol{Bx} + \boldsymbol{g}\ .$$

对任给初始向量 $\boldsymbol{x}^{(0)}$，由迭代公式

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{Bx}^{(m)} + \boldsymbol{g}$$

得到向量序列 $\{\boldsymbol{x}^{(m)}\}$，当迭代序列 $\{\boldsymbol{x}\}$ 收敛于解向量 $\boldsymbol{\alpha}$ 时，可用上面的算法求解．

和解方程的逐步迭代法不同，解线性方程组的迭代法要求按照指定途径由 $\boldsymbol{Ax} = \boldsymbol{b}$ 转换成 $\boldsymbol{x} = \boldsymbol{Bx} + \boldsymbol{g}$ 是唯一的，即两者是一对一的．

和解方程的迭代法类似，本章需要讨论如何建立迭代公式、迭代公式的收敛条件，以及当向量序列 $\{\boldsymbol{x}^{(m)}\}$ 收敛时怎样估计误差．

本章主要介绍最常用的迭代法：简单迭代法、Seidel 迭代法和松弛法．为了研究迭代法的收敛性，先介绍一些向量序列和矩阵序列的极限知识．

4.1　向量序列和矩阵序列的极限

【定义 1】 若 $\{\boldsymbol{x}^{(m)}\}$ 是任一向量序列，$\boldsymbol{x}^{(m)} = (x_1^{(m)}, x_2^{(m)}, \cdots, x_n^{(m)})^{\mathrm{T}}$，对 $i = 1, 2, \cdots, n$ 都有

$$\lim_{m\to\infty} x_i^{(m)} = x_i^*\ ,$$

则称向量序列 $\{\boldsymbol{x}^{(m)}\}$ 收敛于 $\boldsymbol{x}^*$，记为

$$\lim_{m\to\infty} \boldsymbol{x}^{(m)} = \boldsymbol{x}^*\ .$$

【定理 1】 向量序列 $\{\boldsymbol{x}^{(m)}\}$ 依下标收敛于 $\boldsymbol{x}^*$ 的充要条件是向量序列依范数收敛于 $\boldsymbol{x}^*$，即

$$\lim_{m\to\infty} \left\| \boldsymbol{x}^{(m)} - \boldsymbol{x}^* \right\| = 0\ .$$

【证明】 因为

$$\lim_{m\to\infty} x_i^{(m)} = x_i^*,\quad i=1,2,\cdots,n,$$

所以

$$\left\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*\right\|_\infty = \max_i\left|x_i^{(m)}-x_i^*\right| \to 0,$$

由向量范数的等价性有

$$\left\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*\right\| \to 0,\ (\ m\to\infty\ ),$$

由向量序列$\{\boldsymbol{x}^{(m)}\}$依范数收敛于$\boldsymbol{x}^*$，可得

$$\left\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*\right\|_\infty = \max_i\left|x_i^{(m)}-x_i^*\right| \to 0,\ (\ m\to\infty\ ),$$

于是可得

$$\lim_{m\to\infty} x_i^{(m)} = x_i^*,\quad i=1,2,\cdots,n,$$

即向量序列$\{\boldsymbol{x}^{(m)}\}$依下标收敛于$\boldsymbol{x}^*$.

【定义 2】设$\{\boldsymbol{A}^{(m)}\}$是任一矩阵序列，$\boldsymbol{A}^{(m)}=\{a_{ij}^{(m)}\}_{n\times n}$（$m=0,1,\cdots$）是$n$阶方阵，任意元素序列$\{a_{ij}^{(m)}\}$都存在极限

$$\lim_{m\to\infty} a_{ij}^{(m)} = a_{ij}^*,$$

则称矩阵$\boldsymbol{A}^*$为矩阵序列$\{\boldsymbol{A}^{(m)}\}$的极限，或称矩阵序列收敛于矩阵$\boldsymbol{A}^*$，记为

$$\lim_{m\to\infty}\left\|\boldsymbol{A}^{(m)}-\boldsymbol{A}^*\right\| = 0.$$

【定理 2】矩阵序列$\{\boldsymbol{A}^{(m)}\}$收敛于矩阵$\boldsymbol{A}^*$的充分必要条件是矩阵序列$\{\boldsymbol{A}^{(m)}\}$收敛于$\boldsymbol{A}^*$，即

$$\lim_{m\to\infty}\left\|\boldsymbol{A}^{(m)}-\boldsymbol{A}^*\right\| = 0.$$

证明略.

4.2　Jacobi 迭代法

Jacobi 迭代法也称简单迭代法.

4.2.1　Jacobi 迭代法推导

对于方程组

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2 \\ \qquad\qquad\vdots \\ a_{i1}x_1+a_{i2}x_2+\cdots+a_{in}x_n=b_i \\ \qquad\qquad\vdots \\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=b_n \end{cases}$$

将上式第 i（$i=1,2,\cdots,n$）个方程的左端只保留 $a_{ii}x_i$，其余都移到右端得

$$\begin{cases} a_{11}x_1 = b_1 - a_{12}x_2 - \cdots - a_{1n}x_n \\ a_{22}x_2 = b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n \\ \qquad\vdots \\ a_{ii}x_i = b_i - a_{i1}x_1 - \cdots - a_{ii-1}x_{i-1} - a_{ii+1}x_{i+1} - \cdots - a_{in}x_n \\ \qquad\vdots \\ a_{nn}x_n = b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{nn-1}x_{n-1} \end{cases}$$

当 $a_{ii}\neq 0\ (i=1,2,\cdots,n)$时，将上面的方程组中第 $i\ (i=1,2,\cdots,n)$个方程两端同除以 a_{ii} 得

$$\begin{cases} x_1 = \left(b_1 - a_{12}x_2 - \cdots - a_{1n}x_n\right)\big/a_{11} \\ x_2 = \left(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n\right)\big/a_{22} \\ \qquad\vdots \\ x_i = \left(b_i - a_{i1}x_1 - \cdots - a_{ii-1}x_{i-1} - a_{ii+1}x_{i+1} - \cdots - a_{in}x_n\right)\big/a_{ii} \\ \qquad\vdots \\ x_n = \left(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{nn-1}x_{n-1}\right)\big/a_{nn} \end{cases} \tag{4.2-1}$$

式（4.2-1）也可写成

$$x_i = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j\right)\Big/a_{ii}\ ,\quad i=1,2,\cdots,n \tag{4.2-2}$$

和逐次迭代法一样，再将式（4.2-2）改写成

$$\begin{cases} x_i^{(m+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\right)\Big/a_{ii} \qquad m=0,1,\cdots \\ \qquad\quad = b_i^* - \sum_{j=1}^{i-1} a_{ij}^* x_j^{(m)} - \sum_{j=i+1}^{n} a_{ij}^* x_j^{(m)} \qquad ,\quad i=1,2,\cdots,n \\ b_i^* = b_i/a_{ii} \qquad\qquad \left\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\right\| > \varepsilon \\ a_{ij}^* = a_{ij}/a_{ii} \end{cases} \tag{4.2-3}$$

和逐次迭代法不同，式（4.2-1）和式（4.2-3）是一对一的，且 $\boldsymbol{x}^{(0)}$ 可以是任意的.

由于式（4.2-1）、式（4.2-2）变换简单直观，所以 Jacobi 迭代法也称为简单迭代法. 式（4.2-3）就是 Jacobi 迭代法的计算公式，式中 $\left\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\right\| > \varepsilon$ 为迭代条件，ε 为迭代精度，也就是用户给的误差限.

$\left\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\right\| \leqslant \varepsilon$ 为迭代结束条件，显然迭代条件 ＝NOT（迭代结束条件）.

4.2.2 Jacobi 迭代法的矩阵形式

将系数矩阵 $\boldsymbol{A}$ 分解成一个严格下三角矩阵 $-\boldsymbol{L}$ 加上一个对角线矩阵 $\boldsymbol{D}$ 再加上一个严格上三角矩阵 $-\boldsymbol{U}$ 之和，即

$$\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U} \tag{4.2-4}$$

$$-\boldsymbol{L}=\begin{bmatrix} 0 & & & & \\ a_{21} & 0 & & \boldsymbol{0} & \\ a_{31} & a_{32} & 0 & & \\ \cdots & \cdots & \cdots & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn-1} & 0 \end{bmatrix},\quad -\boldsymbol{U}=\begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ & 0 & a_{23} & \cdots & a_{2n} \\ & & \ddots & \cdots & \cdots \\ & \boldsymbol{0} & & 0 & a_{n-1n} \\ & & & & 0 \end{bmatrix},\quad \boldsymbol{D}=\begin{bmatrix} a_{11} & & & \\ & a_{22} & \boldsymbol{0} & \\ & \boldsymbol{0} & \ddots & \\ & & & a_{nn} \end{bmatrix}$$

利用式（4.2-4）可将式（4.2-3）表述成

$$\begin{aligned} \boldsymbol{x}&=\boldsymbol{D}^{-1}\left(\boldsymbol{L}+\boldsymbol{U}\right)\boldsymbol{x}+\boldsymbol{D}^{-1}\boldsymbol{b} \\ &=\boldsymbol{B}_1\boldsymbol{x}+\boldsymbol{g}_1 \end{aligned} \tag{4.2-5}$$

而式（4.2-5）可改写成

$$\boldsymbol{x}^{(m+1)}=\boldsymbol{B}_1\boldsymbol{x}^{(m)}+\boldsymbol{g}_1 \tag{4.2-6}$$

具体计算时，只能用式（4.2-3），而要研究和讨论收敛性时需要用矩阵形式.

4.2.3　Jacobi 迭代法的计算过程和计算实例

1．计算过程

（1）将方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 按式（4.2-3）改写成具体计算公式；

（2）取 $\boldsymbol{x}^{(0)}=(0,0,\cdots,0)^{\mathrm{T}}$：

① 按所给计算公式计算 $\boldsymbol{x}^{(m+1)}\,(m=1,2,\cdots)$；

② 计算 $S=\sum_{i=1}^{n}\left|x_i^{(m+1)}-x_i^{(m)}\right|$，当 $S\leqslant\varepsilon$ 时计算结束.

2．计算实例

【例 1】用 Jacobi 迭代法解方程组

$$\begin{cases} 5x_1-x_2-2x_3=2 \\ -x_1+8x_2-x_3=6 \\ -2x_1-x_2+10x_3=7 \end{cases}$$

要求迭代精度 $\varepsilon=0.02$.

【解】取 $\boldsymbol{x}^{(0)}=(0,0,0)^{\mathrm{T}}$，按式（4.2-3）有

$$\begin{cases} x_1^{(m+1)}=\dfrac{2}{5}+\dfrac{1}{5}x_2^{(m)}+\dfrac{2}{5}x_3^{(m)} \\ x_2^{(m+1)}=\dfrac{3}{4}+\dfrac{1}{8}x_1^{(m)}+\dfrac{1}{8}x_3^{(m)} \\ x_3^{(m+1)}=\dfrac{7}{10}+\dfrac{2}{7}x_1^{(m)}+\dfrac{1}{7}x_2^{(m)} \\ \boldsymbol{x}^{(0)}=(0,0,0)^{\mathrm{T}} \end{cases},\quad \left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m)}\right\|>0.2\times10^{-2},\quad m=1,2,\cdots$$

计算结果如表 4.1 所示.

表 4.1 Jacobi 法的迭代结果

m	0	1	2	3	4	5	6
$x_1^{(m)}$	0.0000	0.4000	0.8300	0.9195	0.9479	0.9888	0.9961
$x_2^{(m)}$	0.0000	0.7500	0.8875	0.9606	0.9843	0.9942	0.9978
$x_3^{(m)}$	0.0000	0.7000	0.8550	0.9548	0.9800	0.9932	0.9972
S		1.8500	0.7225	0.2624	0.0773	0.0821	0.0112

所以用 Jacobi 迭代法的计算结果为

$$x_1 = 0.9961,\quad x_2 = 0.9978,\quad x_3 = 0.9972.$$

和解方程的迭代法不同，解方程组的迭代法对 $\boldsymbol{x}^{(0)}$ 无要求，只要迭代法收敛，无论取 $\boldsymbol{x}^{(0)}$ 为什么，迭代都收敛.

4.3 Seidel 迭代法

Seidel 迭代法也称为 Gauss-Seidel 迭代法.

4.3.1 Seidel 迭代算法推导

由式（4.2-3）可知，在计算 $x_i^{(m+1)}$ 时，$x_1^{(m+1)}$，$x_2^{(m+1)},\cdots,x_{i-1}^{(m+1)}$ 已经算出，一般情况下 $x_1^{(m+1)}$，$x_2^{(m+1)},\cdots,x_{i-1}^{(m+1)}$ 应比 $x_1^{(m)}$，$x_2^{(m)},\cdots,x_{i-1}^{(m)}$ 更接近解 α_1，$\alpha_2,\cdots,\alpha_{i-1}$．在计算 $x_i^{(m+1)}$ 时，用刚计算出的 $x_1^{(m+1)}$，$x_2^{(m+1)},\cdots,x_{i-1}^{(m+1)}$ 代替 $x_1^{(m)}$，$x_2^{(m)}$，$\cdots$，$x_{i-1}^{(m)}$，则变成了 Seidel 迭代公式：

$$\begin{cases} x_i^{(m+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\right) \Big/ a_{ii} \\ \qquad = b_i^* - \sum_{j=1}^{i-1} a_{ij}^* x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}^* x_j^{(m)} \\ \left\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\right\| > \varepsilon \end{cases} \quad \begin{matrix} m = 0,1,\cdots \\ , \\ i = 1,2,\cdots,n \end{matrix} \tag{4.3-1}$$

从式（4.3-1）可以看出，对于确定的 $\boldsymbol{A}$，式（4.3-1）也是确定的，也就是式（4.3-1）是唯一的.

4.3.2 Seidel 迭代法的矩阵表示

将 $\boldsymbol{A}$ 仍分解成 $-\boldsymbol{L}$，$\boldsymbol{D}$ 和 $-\boldsymbol{U}$，它们的意义同前．由式（4.3-1）可以看出

$$\begin{matrix} \boldsymbol{x}^{(m+1)} = \boldsymbol{D}^{-1}(\boldsymbol{L}\boldsymbol{x}^{(m+1)} + \boldsymbol{U}\boldsymbol{x}^{(m)} + \boldsymbol{b}) \\ \left\|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)}\right\| > \varepsilon \end{matrix} \quad , \quad m = 1,2,\cdots$$

由此可推出

$$\begin{cases} \boldsymbol{x}^{(m+1)} = (\boldsymbol{D}+\boldsymbol{L})^{-1}\boldsymbol{U}\boldsymbol{x}^{(m)} + (\boldsymbol{D}+\boldsymbol{L})^{-1}\boldsymbol{b} \\ \quad\quad = \boldsymbol{B}_2\boldsymbol{x}^{(m)} + \boldsymbol{g}_2 \quad\quad , \quad m = 1,2,\cdots \\ \left\| \boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)} \right\| < \varepsilon \end{cases} \tag{4.3-2}$$

4.3.3　Seidel 迭代法的计算过程和计算实例

1．计算过程

（1）按式（4.3-1）写出具体计算公式；

（2）给出 $\boldsymbol{x}^{(0)}$（不妨取 $\boldsymbol{x}^{(0)} = (0,0,\cdots,0)^{\mathrm{T}}$）：

① 按具体计算公式计算 $\boldsymbol{x}^{(m+1)}$（$m = 1,2,\cdots$）；

② 计算 $S = \left\| \boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)} \right\|_1 = \sum_{i=1}^{n} \left| x_i^{(m+1)} - x_i^{(m)} \right|$，当 $S \leqslant \varepsilon$ 时计算结束.

2．计算实例

【例 1】用 Seidel 迭代法解方程组

$$\begin{cases} 5x_1 - x_2 - 2x_3 = 2 \\ -x_1 + 8x_2 - x_3 = 6 \\ -2x_1 - x_2 + 10x_3 = 7 \end{cases}$$

要求迭代精度 $\varepsilon = 0.02$.

【解】按式（4.3-1）有

$$\begin{cases} x_1^{(m+1)} = \dfrac{2}{5} + \dfrac{1}{5}x_2^{(m)} + \dfrac{2}{5}x_3^{(m)} \\ x_2^{(m+1)} = \dfrac{3}{4} + \dfrac{1}{8}x_1^{(m+1)} + \dfrac{1}{8}x_3^{(m)} \\ x_3^{(m+1)} = \dfrac{7}{10} + \dfrac{1}{5}x_1^{(m+1)} + \dfrac{1}{10}x_2^{(m+1)} \\ \left\| \boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)} \right\| > 0.02 \end{cases}$$

取 $\boldsymbol{x}^{(0)} = (0,0,0)^{\mathrm{T}}$，计算结果如表 4.2 所示.

表 4.2　Seidel 迭代法的计算结果

m	0	1	2	3	4	5
$x_1^{(m)}$	0.0000	0.40000	0.90400	0.98524	0.99771	0.99959
$x_2^{(m)}$	0.0000	0.80000	0.97050	0.99539	0.99929	0.99942
$x_3^{(m)}$	0.0000	0.86000	0.97785	0.99659	0.99947	0.99986
S		2.06000	0.79235	0.08338	0.06074	0.00240

计算结果为

$$x_1 = 0.99959\ ,\quad x_2 = 0.99942\ ,\quad x_3 = 0.99986\ .$$

从表 4.2 可知，用 Seidel 迭代法解方程组比 Jacobi 迭代法收敛快（但这不是定理）.

4.4 松弛法

松弛法是 Seidel 迭代法的加权平均，通常松弛法的收敛速度高于 Jacobi 迭代法和 Seidel 迭代法的收敛速度．对于大多数算例，松弛法与 Jacobi 迭代法和 Seidel 迭代法相比，耗费的机时都要少.

4.4.1 松弛法计算公式

Seidel 迭代公式可以改写为

$$\begin{aligned} x_i^{(m+1)} &= x_i^{(m)} + \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(m)} \right) \\ &= x_i^{(m)} + r_i^{(m+1)} \end{aligned} \tag{4.4-1}$$

其中，$r_i^{(m+1)}$ 可认为是第 $m+1$ 步对第 m 步的 $x_i^{(m)}$ 的修正量．若迭代法收敛，则 $\lim\limits_{m\to\infty} r_i^{(m+1)} = 0$ ．因此，对于 Seidel 迭代法的迭代过程，我们可以认为它的第 $m+1$ 步计算实际上是对第 m 步各分量增加一个修正量 $r_i^{(m+1)}$ ．为了提高 Seidel 迭代法的收敛速度，可对修正量乘上松弛因子 $\tilde{\omega}$，即

$$\begin{aligned} x_i^{(m+1)} &= x_i^{(m)} + \frac{\tilde{\omega}}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(m)} \right) \\ &= (1-\tilde{\omega})x_i^{(m)} + \frac{\tilde{\omega}}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)} \right) \end{aligned} \quad \begin{array}{l} m = 1,2,\cdots \\ i = 1,2,\cdots,n \end{array} \tag{4.4-2}$$

将式（4.4-2）加上迭代条件 $\left\| \boldsymbol{x}^{(m+1} - \boldsymbol{x}^{(m)} \right\| > \varepsilon$ 并取 $\boldsymbol{x}^{(0)} = (0,0,\cdots,0)^{\mathrm{T}}$ ，则完成了松弛法迭代公式的推导.

式（4.4-2）中，当 $\tilde{\omega} = 1$ 时，则变成了 Seidel 迭代公式；当 $0 < \tilde{\omega} < 1$ 时，称为亚松弛迭代；当 $1 < \tilde{\omega} < 2$ 时，称为超松弛迭代.

到底取何值，通常要凭经验，一般情况下使用超松弛法．若 Seidel 迭代法收敛，那么用超松弛法肯定收敛且收敛速度会高于 Seidel 迭代法.

4.4.2 松弛法的矩阵形式

用 a_{ii} 乘上式（4.4-2）并整理得

$$a_{ii}x_i^{(m+1)} + \tilde{\omega}\sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} = (1-\tilde{\omega})a_{ii}x_i^{(m)} - \tilde{\omega}\sum_{j=i+1}^{n} a_{ij}x_j^{(m)} + \tilde{\omega}b_i\ ,$$

由此得

$$(\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})\boldsymbol{x}^{(m+1)}=((1-\tilde{\omega})\boldsymbol{D}+\tilde{\omega}\boldsymbol{U})\boldsymbol{x}^{(m)}+\tilde{\omega}\boldsymbol{b}$$

$$\begin{cases}\boldsymbol{x}^{(m+1)}=(\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})^{-1}((1-\tilde{\omega})\boldsymbol{D}+\tilde{\omega}\boldsymbol{U})\boldsymbol{x}^{(m)}+\tilde{\omega}(\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})^{-1}\boldsymbol{b}\\ \quad=\boldsymbol{B}_{\tilde{\omega}}\boldsymbol{x}^{(m)}+\boldsymbol{g}_{\tilde{\omega}}\\ \left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m)}\right\|>\varepsilon\end{cases}\tag{4.4-3}$$

4.4.3　松弛法的计算过程和计算实例

1．计算过程

（1）凭经验给出松弛因子 $\tilde{\omega}$；

（2）按式（4.4-2）给出具体的松弛法计算公式；

（3）给出 $\boldsymbol{x}^{(0)}$（不妨取 $\boldsymbol{x}^{(0)}=(0,0,\cdots,0)^{\mathrm{T}}$）：

① 按具体计算公式计算 $\boldsymbol{x}^{(m+1)}$（$m=1,2,\cdots$）；

② 计算 $S=\sum_{i=1}^{n}\left|x_i^{(m+1)}-x_i^{(m)}\right|$，当 $S\leqslant\varepsilon$ 时计算结束.

2．计算实例

【例 1】分别用 Jacobi 迭代法、Seidel 迭代法和松弛法解如下线性代数方程组，迭代精度 $\varepsilon=0.02$，松弛因子 $\tilde{\omega}=1.25$：

$$\begin{cases}x_1+0.8x_2=0.2\\ 0.8x_1+x_2=-0.2\end{cases},\quad \left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m)}\right\|>0.02,\quad m=0,1,\cdots$$

【解】Jacobi 法的迭代公式为

$$\begin{cases}x_1^{(m+1)}=-0.8x_2^{(m)}+0.2\\ x_2^{(m+1)}=-0.8x_1^{(m)}-0.2\end{cases},\quad \left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m)}\right\|>0.02,\quad m=0,1,\cdots$$

Seidel 法的迭代公式为

$$\begin{cases}x_1^{(m+1)}=-0.8x_2^{(m)}+0.2\\ x_2^{(m+1)}=-0.8x_1^{(m+1)}-0.2\end{cases},\quad \left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m)}\right\|>0.02$$

松弛法的迭代公式为

$$\begin{cases}x_1^{(m+1)}=-0.25x_1^{(m)}+x_2^{(m)}+0.25\\ x_2^{(m+1)}=-x_1^{(m+1)}-0.25x_2^{(m)}-0.25\end{cases},\quad \left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m)}\right\|>0.02$$

都取 $\boldsymbol{x}^{(0)}=(0,0)^{\mathrm{T}}$，计算结果如表 4.3 所示.

表 4.3 三种算法的计算结果

m	Jacobi		Seidel		松驰法($\tilde{\omega}=1.25$)	
	$x_1^{(m)}$	$x_2^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$
0	0.00000	0.00000	0.00000	0.0000	0.00000	0.00000
1	0.20000	−0.20000	0.20000	−0.36000	0.25000	−0.50000
2	0.36000	−0.36000	0.48800	−0.59040	0.68750	−0.81250
3	0.48800	−0.48800	0.67232	−0.73786	0.89062	−0.93750
4	0.59040	−0.59040	0.79228	−0.83223	0.96408	−0.98047
5	0.67232	−0.67232	0.86583	−0.89296	0.98944	−0.99433
6	0.73786	−0.73786	0.91413	−0.93130	0.99169	−0.99574
7	0.79028	−0.79028				
8	0.83228	−0.83228				

用松弛法计算的结果为

$$x_1 = 0.99169\ ,\quad x_2 = 0.99574\ .$$

从计算结果可以看出：松弛法收敛最快，迭代 6 次就达到了计算精度；Jacobi 迭代法经过 8 次迭代尚未达到计算精度；而 Seidel 迭代法经 6 次迭代也未达到计算精度，这里因篇幅关系未写出最后结果.

4.5 迭代法收敛条件

为了保证迭代法的有效性，必须要求迭代过程是收敛的．因为一个发散的迭代过程，无论进行多少次迭代，其计算过程都是毫无价值的．因此，必须讨论迭代法的收敛条件.

4.5.1 对角占优矩阵和不可约矩阵

【定理 1】若 n 阶方阵 $\boldsymbol{A}$ 是严格对角占优的，则 $\boldsymbol{A}$ 是非奇异的.

【证明】将 $\boldsymbol{A}$ 分解成 $\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$，其中 $\boldsymbol{D}$，$-\boldsymbol{L}$ 和 $-\boldsymbol{U}$ 分别为 $\boldsymbol{A}$ 的对角阵、严格下三角阵和严格上三角阵．由于 $\boldsymbol{A}$ 严格对角占优，因此 $a_{ii}\neq 0$（$i=1,2,\cdots,n$），从而 $\boldsymbol{D}$ 为非奇异矩阵，且

$$\left\|\boldsymbol{B}_1\right\|_\infty = \left\|\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})\right\|_\infty = \max \sum_{\substack{j=1\\ j\neq i}}^{n}\left|\frac{a_{ij}}{a_{ii}}\right| < 1\ .$$

由此可知：$\boldsymbol{I}-\boldsymbol{B}_1=\boldsymbol{I}-\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})$ 是非奇异矩阵，从而 $\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}=\boldsymbol{D}(\boldsymbol{I}-\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U}))$ 为非奇异矩阵.

【定义 1】对于阶数不小于 2 的方阵 $\boldsymbol{A}$，若经过行的互换和相应列的互换后，可以化为

$$\begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{A}_{22} \end{bmatrix} \tag{4.5-1}$$

其中，$\boldsymbol{A}_{11}$ 和 $\boldsymbol{A}_{22}$ 都是阶数不小于 1 的方阵，$\boldsymbol{0}$ 为零矩阵，则称方阵 $\boldsymbol{A}$ 是可约的；否则，称为不可约的.

【例 1】设方阵 $\boldsymbol{A}$ 为

$$\begin{bmatrix} 1 & 2 & -3 & 4 & 5 \\ 0 & 6 & 0 & 7 & 0 \\ 8 & 9 & 10 & -1 & -2 \\ 0 & -3 & 0 & -4 & 0 \\ 5 & 0 & 7 & 8 & 9 \end{bmatrix}$$

验证方阵是可约的.

进行行列交换：先分别交换 $\boldsymbol{A}$ 的第 1 行、第 3 行和第 1 列、第 3 列，再将所得矩阵分别交换第 2 行、第 5 行和第 2 列、第 5 列，可将 $\boldsymbol{A}$ 化为

$$\begin{bmatrix} 10 & -2 & 8 & -1 & 8 \\ 7 & 9 & 5 & 8 & 0 \\ -3 & 5 & 1 & 4 & 2 \\ 0 & 0 & 0 & -4 & -3 \\ 0 & 0 & 0 & 7 & 6 \end{bmatrix}$$

由此可知 $\boldsymbol{A}$ 是可约的.

【定理 2】若 n（$n \geqslant 2$）阶方阵 $\boldsymbol{A}$ 不可约且弱对角占优，则 $\boldsymbol{A}$ 为非奇异矩阵.

证明略.

4.5.2　迭代法的收敛条件和误差估计

【定理 3】设 $\boldsymbol{A}$ 为 n 阶方阵，则 $\lim\limits_{m\to\infty} \boldsymbol{A}^m = 0$ 的充要条件是 $\rho(\boldsymbol{A}) < 1$.

【证明】必要性

若 $\lim\limits_{m\to\infty} \boldsymbol{A}^m = 0$，由 4.1 节的定理 2 有 $\lim\limits_{m\to\infty} \left\| \boldsymbol{A}^m \right\| = 0$，而

$$0 \leqslant \rho(\boldsymbol{A}^m) = \left(\rho(\boldsymbol{A})\right)^m \leqslant \left\| \boldsymbol{A}^m \right\| \to 0,$$

从而有

$$\lim_{m\to\infty} \rho((\boldsymbol{A}))^m = 0,$$

所以

$$\rho(\boldsymbol{A}) < 1.$$

充分性

若 $\rho(\boldsymbol{A}) < 1$，取 $\varepsilon = \dfrac{1-\rho(\boldsymbol{A})}{2} > 0$，由谱半径和范数关系有：存在某种矩阵范数 $\left\| \bullet \right\|_*$，使得

$$\|\boldsymbol{A}\|_* \leqslant \rho(\boldsymbol{A}) + \varepsilon = \frac{1+\rho(\boldsymbol{A})}{2} < 1 .$$

又

$$\left\|\boldsymbol{A}^m\right\|_* \leqslant \|\boldsymbol{A}\|_*^m ,$$

于是

$$0 \leqslant \lim_{m\to\infty} \left\|\boldsymbol{A}^m\right\|_* \leqslant \lim_{m\to\infty} \|\boldsymbol{A}\|_*^m = 0 ,$$

从而

$$\lim_{m\to\infty} \left\|\boldsymbol{A}^m\right\|_* = 0 .$$

根据 4.1 节的定理 2 可知 $\lim\limits_{m\to\infty} \boldsymbol{A}^m = 0$.

【定理 4】对任意初始向量 $\boldsymbol{x}^{(0)}$，由迭代公式

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{B}\boldsymbol{x}^{(m)} + \boldsymbol{g} , \quad m = 0,1,\cdots \tag{4.5-2}$$

所得的序列 $\{\boldsymbol{x}^{(m)}\}$ 收敛于方程 $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{x} + \boldsymbol{g}$ 的唯一解 $\boldsymbol{x}^*$ 的充要条件是，迭代矩阵的谱半径

$$\rho(\boldsymbol{B}) < 1 \tag{4.5-3}$$

【证明】必要性

对任意初始向量 $\boldsymbol{x}^{(0)}$，由迭代公式（4.5-2）得到的序列 $\{\boldsymbol{x}^{(m)}\}$ 收敛于 $\boldsymbol{x}^*$，即

$$\lim_{m\to\infty} \boldsymbol{x}^{(m)} = \boldsymbol{x}^* .$$

在式（4.5-2）两边取极限得

$$\boldsymbol{x}^* = \boldsymbol{B}\boldsymbol{x}^* + \boldsymbol{g} ,$$

将此式与 $\boldsymbol{x}^{(m)} = \boldsymbol{B}\boldsymbol{x}^{(m-1)} + \boldsymbol{g}$ 两边相减，得

$$\begin{aligned} \boldsymbol{x}^{(m)} - \boldsymbol{x}^* &= \boldsymbol{B}(\boldsymbol{x}^{(m-1)} - \boldsymbol{x}^*) \\ &= \boldsymbol{B}(\boldsymbol{B}\boldsymbol{x}^{(m-2)} + \boldsymbol{g} - (\boldsymbol{B}\boldsymbol{x}^* + \boldsymbol{g})) \\ &= \boldsymbol{B}^2(\boldsymbol{x}^{(m-2)} - \boldsymbol{x}^*) \\ &\cdots \\ &= \boldsymbol{B}^m(\boldsymbol{x}^{(0)} - \boldsymbol{x}^*) \end{aligned}$$

两边取极限，则有

$$\boldsymbol{B}^m(\boldsymbol{x}^{(0)} - \boldsymbol{x}^*) \to 0 \ (m \to \infty).$$

由 $\boldsymbol{x}^{(0)}$ 的任意性，总可以取 $\boldsymbol{x}^{(0)}$，使得 $\boldsymbol{x}^{(0)} - \boldsymbol{x}^*$ 不等于 0，所以

$$\lim_{m\to\infty} \boldsymbol{B}^m = 0 .$$

由定理 3 得

$$\rho(\boldsymbol{B})<1\,.$$

充分性

设 $\rho(\boldsymbol{B})<1$，则 $\lambda=1$ 不是 $\boldsymbol{B}$ 的特征值，于是行列式 $|\boldsymbol{B}-\boldsymbol{I}|\neq 0$，从而方程组 $\boldsymbol{x}=\boldsymbol{B}\boldsymbol{x}+\boldsymbol{g}$ 有唯一解 $\boldsymbol{x}^*=\boldsymbol{B}\boldsymbol{x}^*+\boldsymbol{g}$．于是可得

$$\boldsymbol{x}^{(m)}-\boldsymbol{x}^*=\boldsymbol{B}^m(\boldsymbol{x}^{(0)}-\boldsymbol{x}^*)\,,$$

所以，对任意初始向量 $\boldsymbol{x}^{(0)}$，有

$$\lim_{m\to\infty}(\boldsymbol{x}^{(m)}-\boldsymbol{x}^*)=\lim_{m\to\infty}\boldsymbol{B}^m(\boldsymbol{x}^{(0)}-\boldsymbol{x}^*)\,.$$

由于 $\rho(\boldsymbol{B})<1$，由定理 3 可得 $\lim\limits_{m\to\infty}\boldsymbol{B}^m=0$．从而

$$\lim_{m\to\infty}\boldsymbol{x}^{(m)}=\boldsymbol{x}^*\,,$$

即由迭代公式 $\boldsymbol{x}^{(m+1)}=\boldsymbol{B}\boldsymbol{x}^{(m)}+\boldsymbol{g}$ 产生的序列 $\{\boldsymbol{x}^{(m)}\}$ 收敛于方程组 $\boldsymbol{x}=\boldsymbol{B}\boldsymbol{x}+\boldsymbol{g}$ 的唯一解 $\boldsymbol{x}^*$．

定理 4 说明，迭代法的收敛与否仅与迭代矩阵的谱半径有关，而与初始向量 $\boldsymbol{x}^{(0)}$ 和方程组的右端项 $\boldsymbol{b}$ 无关.

定理 4 有重要的理论意义，但由于谱半径 $\rho(\boldsymbol{B})$ 求解困难，下面给出实用的收敛充分条件.

【定理 5】若迭代矩阵 $\boldsymbol{B}$ 满足

$$\|\boldsymbol{B}\|<1 \tag{4.5-4}$$

则对任意初始向量 $\boldsymbol{x}^{(0)}$，由迭代公式（4.5-2）得到的序列收敛于 $\boldsymbol{x}=\boldsymbol{B}\boldsymbol{x}+\boldsymbol{g}$ 的唯一解 $\boldsymbol{x}^*$，且有误差估计式

$$\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*\|\leqslant\frac{\|\boldsymbol{B}\|}{1-\|\boldsymbol{B}\|}\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^{(m-1)}\| \tag{4.5-5}$$

及

$$\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*\|\leqslant\frac{\|\boldsymbol{B}\|^m}{1-\|\boldsymbol{B}\|}\|\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)}\| \tag{4.5-6}$$

【证明】由 $\rho(\boldsymbol{B})\leqslant\|\boldsymbol{B}\|<1$ 和定理 4 可知：对任意初始向量 $\boldsymbol{x}^{(0)}$，由迭代公式 $\boldsymbol{x}^{(m+1)}=\boldsymbol{B}\boldsymbol{x}^{(m)}+\boldsymbol{g}$ 得到的向量序列 $\{\boldsymbol{x}^{(m)}\}$ 收敛于方程组 $\boldsymbol{x}=\boldsymbol{B}\boldsymbol{x}+\boldsymbol{g}$ 的唯一解 $\boldsymbol{x}^*$，从而有

$$\boldsymbol{x}^{(m)}-\boldsymbol{x}^*=\boldsymbol{B}(\boldsymbol{x}^{(m-1)}-\boldsymbol{x}^*)\,,$$

两边取范数并利用范数性质，可得

$$\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*\|\leqslant\|\boldsymbol{B}\|\|\boldsymbol{x}^{(m-1)}-\boldsymbol{x}^*\|\leqslant\|\boldsymbol{B}\|\left(\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^{(m-1)}\|+\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*\|\right),$$

移项并整理，即得误差估计式（4.5-5）.

又因为

$$\boldsymbol{x}^{(m)}-\boldsymbol{x}^{(m-1)}=\boldsymbol{B}(\boldsymbol{x}^{(m-1)}-\boldsymbol{x}^{(m-2)})=\boldsymbol{B}^2(\boldsymbol{x}^{(m-2)}-\boldsymbol{x}^{(m-3)})=\cdots=\boldsymbol{B}^{m-1}(\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)})\,,$$

所以

$$\left\|\boldsymbol{x}^{(m)}-\boldsymbol{x}^{(m-1)}\right\|\leqslant\left\|\boldsymbol{B}^{m-1}\right\|\left\|\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)}\right\|\leqslant\|\boldsymbol{B}\|^{m-1}\left\|\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)}\right\|.$$

将上式代入式（4.5-5），便得误差估计式（4.5-6）.

通常把式（4.5-5）称为事后估计，而把式（4.5-6）称为事前估计.

有了上面的定理，在实际计算中，若允许误差是ε，只要相邻两次迭代向量差的范数满足关系式

$$\left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m-1)}\right\|\leqslant\frac{1-\|\boldsymbol{B}\|}{\|\boldsymbol{B}\|}\varepsilon \tag{4.5-7}$$

$\boldsymbol{x}^{(m)}$已经是$\boldsymbol{x}^{*}$满足精度要求的近似解向量，那么迭代过程即可停止. 由于$\|\boldsymbol{B}\|$的计算麻烦，故程序中常用$\left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{(m-1)}\right\|\leqslant\varepsilon$作为迭代结束的条件.

另外，根据事前估计和预先给定的允许误差ε，可以求出迭代次数. 因为由

$$\left\|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^{*}\right\|\leqslant\frac{1-\|\boldsymbol{B}\|^{m}}{\|\boldsymbol{B}\|}\left\|\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)}\right\|\leqslant\varepsilon$$

可得

$$\|\boldsymbol{B}\|^{m}\leqslant\frac{\varepsilon(1-\|\boldsymbol{B}\|)}{\left\|\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)}\right\|},$$

由于$\|\boldsymbol{B}\|<1$，所以

$$m\geqslant\frac{\lg\dfrac{\varepsilon(1-\|\boldsymbol{B}\|)}{\left\|\boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)}\right\|}}{\lg\|\boldsymbol{B}\|} \tag{4.5-8}$$

【例 2】讨论解方程组

$$\begin{cases}\ \ x_1-2x_2+2x_3=\ 1\\ -x_1+\ x_2-\ x_3=\ 1\\ -2x_1-2x_2+\ x_3=-3\end{cases}$$

的 Jacobi 迭代法和 Gauss-Seidel 迭代法的收敛性.

【解】对于 Jacobi 迭代法，其迭代矩阵为

$$\boldsymbol{B}_1=\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}^{-1}\begin{bmatrix}0&2&-2\\1&0&1\\2&2&0\end{bmatrix}=\begin{bmatrix}0&2&-2\\1&0&1\\2&2&0\end{bmatrix},$$

其特征方程为

$$\left|\boldsymbol{B}_1-\lambda\boldsymbol{I}\right|=\begin{vmatrix}-\lambda&2&2\\1&-\lambda&1\\2&2&-\lambda\end{vmatrix}=-\lambda^3=0,$$

因为特征值 $\lambda_1=\lambda_2=\lambda_3=0$，所以 $\rho(\boldsymbol{B}_1)=0<1$，由定理 4 可知，Jacobi 迭代法收敛.

对于 Gauss-Seidel 迭代法，其迭代矩阵为

$$\begin{aligned}\boldsymbol{B}_2&=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U}\\&=\begin{bmatrix}1&0&0\\-1&1&0\\-2&-2&1\end{bmatrix}^{-1}\begin{bmatrix}0&2&-2\\0&0&1\\0&0&0\end{bmatrix}\\&=\begin{bmatrix}1&0&0\\1&1&0\\4&2&1\end{bmatrix}\begin{bmatrix}0&2&-2\\0&0&1\\0&0&0\end{bmatrix},\\&=\begin{bmatrix}0&2&-2\\0&2&-1\\0&8&-6\end{bmatrix}\end{aligned}$$

其特征方程为

$$\left|\boldsymbol{B}_2-\lambda\boldsymbol{I}\right|=\begin{vmatrix}-\lambda&2&-2\\0&2-\lambda&-1\\0&8&-6-\lambda\end{vmatrix}=-\lambda(\lambda^2+4\lambda-4)=0.$$

因为特征值 $\lambda_1=0$，$\lambda_2=-2+2\sqrt{2}$，$\lambda_3=-2-2\sqrt{2}$，所以 $\rho(\boldsymbol{B}_2)=2+2\sqrt{2}>1$．由定理 4 可知，Gauss-Seidel 迭代法不收敛.

【例 3】讨论解方程组

$$\begin{cases}x_1+0.5x_2+0.5x_3=2\\0.5x_1+x_2+0.5x_3=2\\0.5x_1+0.5x_2+x_3=2\end{cases}$$

的 Jacobi 迭代法和 Gauss-Seidel 迭代法的收敛性.

【解】对于 Jacobi 迭代法，其迭代矩阵为

$$\boldsymbol{B}_1=\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})=\begin{bmatrix}0&-0.5&-0.5\\-0.5&0&-0.5\\-0.5&-0.5&0\end{bmatrix},$$

其特征方程为

$$\left|\boldsymbol{B}_1-\lambda\boldsymbol{I}\right|=-\lambda^3+0.75\lambda-0.25=-(\lambda+1)(\lambda-0.5)^2=0,$$

所以 $\rho(\boldsymbol{B}_1)=1$，由定理 4 可知，Jacobi 迭代法不收敛.

对于 Gauss-Seidel 迭代法，其迭代矩阵为

$$\boldsymbol{B}_2=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U}=\begin{bmatrix}0 & -0.5 & -0.5\\ 0 & 0.25 & -0.25\\ 0 & 0.125 & 0.375\end{bmatrix},$$

因为其特征方程为 $|\boldsymbol{B}_2-\lambda\boldsymbol{I}|=-\lambda(\lambda^2-0.6254\lambda+0.125)=0$，所以 $\rho(\boldsymbol{B}_2)=0.353\ 553\ 99<1$，由定理 4 可知，Gauss-Seidel 迭代法收敛.

例 2 和例 3 说明，对于同一个线性方程组，Jacobi 迭代法和 Gauss-Seidel 迭代法的收敛与否没有必然联系.

【例 4】设给定方程组的系数矩阵为

$$\boldsymbol{A}=\begin{bmatrix}5 & -1 & -2\\ -1 & 8 & -1\\ -2 & -1 & 10\end{bmatrix},$$

讨论 Jacobi 迭代法和 Gauss-Seidel 迭代法的收敛性.

【解】对于 Jacobi 迭代法，其迭代矩阵为

$$\boldsymbol{B}_1=\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})=\begin{bmatrix}0 & 0.2 & 0.4\\ 0.125 & 0 & 0.125\\ 0.2 & 0.1 & 0\end{bmatrix},$$

因为 $\|\boldsymbol{B}_1\|_\infty=0.6<1$，所以根据定理 5，Jacobi 迭代法收敛.

对于 Gauss-Seidel 迭代法，其迭代矩阵为

$$\boldsymbol{B}_2=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U}=\begin{bmatrix}0 & 0.2 & 0.4\\ 0 & 0.025 & 0.175\\ 0 & 0.0425 & 0.0975\end{bmatrix},$$

因为 $\|\boldsymbol{B}_2\|_\infty=0.6<1$，所以根据定理 5，Gauss-Seidel 迭代法收敛.

定理 4 和定理 5 是针对一般迭代法的收敛性定理. 针对 Jacobi 迭代法、Gauss-Seidel 迭代法及松弛法，将针对一些特殊的系数矩阵给出几个实用的收敛的充分条件.

【定理 6】若系数矩阵 $\boldsymbol{A}$ 是严格对角占优矩阵，则 Jacobi 迭代法和 Gauss-Seidel 迭代法都收敛.

【证明】先证 Jacobi 迭代法收敛. $\boldsymbol{A}$ 是严格对角占优矩阵，即满足

$$|a_{kk}|>\sum_{\substack{j=1\\ j\neq k}}^{n}|a_{kj}|,\quad k=1,2,\cdots,n,$$

可得

$$\|\boldsymbol{B}_1\|_\infty=\max\sum_{\substack{j=1\\ j\neq k}}^{n}\left|\frac{a_{kj}}{a_{kk}}\right|<1,$$

根据定理 5，Jacobi 迭代法收敛.

再证 Gauss-Seidel 迭代法收敛. 根据定理 5，只需证明 $\boldsymbol{B}_2=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U}$ 的某种范数小于 1. 下面证明 $\|\boldsymbol{B}_2\|_\infty<1$.

由矩阵范数定义有

$$\|\boldsymbol{B}_2\|_\infty=\max_{\|\boldsymbol{x}\|_\infty=1}\|\boldsymbol{B}_2\boldsymbol{x}\|_\infty .$$

令 $\boldsymbol{y}=\boldsymbol{B}_2\boldsymbol{x}$，则由 $\boldsymbol{B}_2=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U}$ 可得

$$\boldsymbol{y}=\boldsymbol{D}^{-1}\boldsymbol{L}\boldsymbol{y}+\boldsymbol{D}^{-1}\boldsymbol{U}\boldsymbol{x} .$$

上式第 k 个方程为

$$y_k=\sum_{j=1}^{k-1}\left(-\frac{a_{kj}}{a_{kk}}\right)y_j+\sum_{j=k+1}^{n}\left(-\frac{a_{kj}}{a_{kk}}\right)x_j .$$

令 $|y_k|=\max_i|y_i|$，则

$$\|\boldsymbol{y}\|_\infty=|y_k|\leqslant\sum_{j=1}^{k-1}\left(\frac{a_{kj}}{a_{kk}}\right)\|\boldsymbol{y}\|_\infty+\sum_{j=k+1}^{n}\left(\frac{a_{kj}}{a_{kk}}\right)\|\boldsymbol{x}\|_\infty .$$

令

$$r_i=\sum_{j=1}^{i-1}\left|\frac{a_{ij}}{a_{ii}}\right|,\quad s_i=\sum_{j=i+1}^{n}\left|\frac{a_{ij}}{a_{ii}}\right|,$$

则有

$$\|\boldsymbol{y}\|_\infty\leqslant r_k\|\boldsymbol{y}\|_\infty+s_k\|\boldsymbol{x}\|_\infty .$$

由于 $\boldsymbol{A}$ 严格对角占优，即 $r_i+s_i<1$（$i=1,2,\cdots,n$），于是

$$\|\boldsymbol{y}\|_\infty\leqslant\frac{s_k}{1-r_k}\|\boldsymbol{x}\|_\infty ,$$

从而

$$\|\boldsymbol{B}_2\|_\infty=\max_{\|\boldsymbol{x}\|_\infty=1}\|\boldsymbol{y}\|_\infty\leqslant\max_{1\leqslant k\leqslant n}\frac{s_k}{1-r_k}<1,$$

故 Gauss-Seidel 迭代法收敛.

如在例 4 中，系数矩阵

$$\boldsymbol{A}=\begin{bmatrix}5&-1&-2\\-1&8&-1\\-2&-1&10\end{bmatrix}$$

是严格对角占优的，根据定理 6，Jacobi 迭代法和 Gauss-Seidel 迭代法都收敛.

【定理 7】若系数矩阵 $\boldsymbol{A}$ 不可约且弱对角占优，则 Jacobi 迭代法和 Gauss-Seidel 迭代法都收敛.

【证明】因为矩阵 $\boldsymbol{A}$ 不可约且弱对角占优，根据定理 2，$\boldsymbol{A}$ 非奇异且主对角线元素都不

等于 0．设 $\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$，再根据定理 4，只需证明 $\boldsymbol{B}_1=\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U})$ 和 $\boldsymbol{B}_2=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{U}$ 的谱半径小于 1 即可．

用反证法证明：对于 Jacobi 迭代法，令 $\rho(\boldsymbol{B}_1)=|\lambda_1|$，因为 λ_1 是 $\boldsymbol{B}_1$ 的特征值，所以

$$\det(\lambda_1\boldsymbol{I}-\boldsymbol{D}^{-1}(\boldsymbol{L}+\boldsymbol{U}))=0\ ,$$

从而

$$\det(\lambda_1\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U})=0\ .$$

假设 $\rho(\boldsymbol{B}_1)=|\lambda_1|\geqslant 1$，当 $\boldsymbol{A}$ 不可约且弱对角占优时，$\lambda_1\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$ 仍是不可约且弱对角占优的．根据定理 2，矩阵 $\lambda_1\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$ 非奇异，即 $\det(\lambda_1\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U})\neq 0$，与前面的结论矛盾，所以 $\rho(\boldsymbol{B}_1)=|\lambda_1|<1$，即 Jacobi 迭代法收敛．

类似地，可证明 Gauss-Seidel 迭代法收敛．

在例 4 中，系数矩阵 $\boldsymbol{A}$ 也是不可约且弱对角占优的．根据定理 7，Jacobi 迭代法和 Gauss-Seidel 迭代法都收敛．

【定理 8】若系数矩阵 $\boldsymbol{A}$ 为实对称正定矩阵，则 Gauss-Seidel 迭代法收敛．

【证明】由于矩阵 $\boldsymbol{A}$ 为实对称正定的，则 $\boldsymbol{A}$ 可分解为 $\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{L}^{\mathrm{T}}$，其中 $\boldsymbol{D}$ 为对角阵，$\boldsymbol{L}$ 为严格下三角阵．设 λ 为迭代矩阵 $\boldsymbol{B}_2=(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{L}^{\mathrm{T}}$ 的任一特征值，$\boldsymbol{x}$ 为其对应的特征向量，则有

$$(\boldsymbol{D}-\boldsymbol{L})^{-1}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\lambda\boldsymbol{x}\ ,$$

所以

$$\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\lambda(\boldsymbol{D}-\boldsymbol{L})\boldsymbol{x}=\lambda(\boldsymbol{A}+\boldsymbol{L}^{\mathrm{T}})\boldsymbol{x}\ ,$$

从而

$$(1-\lambda)\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\lambda\boldsymbol{A}\boldsymbol{x}\ .$$

用向量 $\boldsymbol{x}$ 的共轭转置 $\boldsymbol{x}^*$ 左乘以上式的两端，得

$$(1-\lambda)\boldsymbol{x}^*\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\lambda\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\ .$$

由 $\boldsymbol{A}$ 的对称正定性和上式可知 $\lambda\neq 1$，于是

$$\boldsymbol{x}^*\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\frac{\lambda}{1-\lambda}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\ ,$$

取上式两端的共轭转置得

$$x^*\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\frac{\overline{\lambda}}{1-\overline{\lambda}}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\ ,$$

将以上两式相加得

$$\boldsymbol{x}^*(\boldsymbol{L}+\boldsymbol{L}^{\mathrm{T}})\boldsymbol{x}=\left(\frac{\lambda}{1-\lambda}+\frac{\overline{\lambda}}{1-\overline{\lambda}}\right)\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\ .$$

由 $\boldsymbol{L}+\boldsymbol{L}^{T}=\boldsymbol{D}-\boldsymbol{A}$，化简上式得

$$\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}=\left(1+\frac{\lambda}{1-\lambda}+\frac{\overline{\lambda}}{1-\overline{\lambda}}\right)\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}=\frac{1-|\lambda|^2}{|1-\lambda|^2}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\ .$$

因为 $\boldsymbol{A}$ 和 $\boldsymbol{D}$ 均为实对称正定阵，且 $\boldsymbol{x}\neq\boldsymbol{0}$，所以 $\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}>0$，$\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}>0$；又因为 $\lambda\neq1$，所以 $|1-\lambda|>0$，从而由上式可得

$$1-|\lambda|^2>0,$$

即 $\rho(\boldsymbol{B}_2)<1$，这就证明了 Gauss-Seidel 迭代法收敛.

在例 3 的线性方程组中，因为系数矩阵

$$\boldsymbol{A}=\begin{bmatrix}1 & 0.5 & 0.5\\ 0.5 & 1 & 0.5\\ 0.5 & 0.5 & 1\end{bmatrix}$$

是实对称矩阵，且各阶主子式都大于零，所以 $\boldsymbol{A}$ 是实对称正定矩阵. 根据定理 8，Gauss-Seidel 迭代法收敛.

【定理 9】松弛迭代法收敛的必要条件是 $0<\tilde{\omega}<2$.

【证明】若松弛迭代法收敛，则由定理 4 可知 $\rho(\boldsymbol{B}_{\tilde{\omega}})<1$. 设 $\lambda_1,\lambda_2,\cdots,\lambda_n$ 为矩阵 $\boldsymbol{B}_{\tilde{\omega}}$ 的特征值，则

$$\begin{aligned}\left|\det(\boldsymbol{B}_{\tilde{\omega}})\right|&=\det((\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})^{-1})\det((1-\tilde{\omega})\boldsymbol{D}+\tilde{\omega}\boldsymbol{U}))\\&=\frac{1}{a_{11}a_{22}\cdots a_{nn}}\times(1-\tilde{\omega})^n a_{11}a_{22}\cdots a_{nn}\\&=(1-\tilde{\omega})^n\end{aligned}$$

由以上等式得

$$\left|\det(\boldsymbol{B}_{\tilde{\omega}})\right|=\left|(1-\tilde{\omega})^n\right|<1,$$

所以 $\left|(1-\tilde{\omega})^n\right|<1$，即 $0<\omega<2$.

定理 9 说明：若要松弛迭代法收敛，则必须选取松弛因子 $\tilde{\omega}$，使 $\tilde{\omega}\in(0,2)$. 但是，当 $\tilde{\omega}\in(0,2)$ 时，对一般线性方程组，松弛迭代法未必都收敛. 下面给出松弛迭代法收敛的一些充分条件.

【定理 10】若系数矩阵 $\boldsymbol{A}$ 为实对称正定矩阵，则当 $0<\tilde{\omega}<2$ 时，松弛迭代法收敛.

【证明】设 λ 是迭代矩阵 $\boldsymbol{B}_{\tilde{\omega}}=(\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})^{-1}((1-\tilde{\omega})\boldsymbol{D}+\tilde{\omega}\boldsymbol{U})$ 的任意特征值，$\boldsymbol{x}$ 是相应的特征向量. 由于 $\boldsymbol{A}$ 为对称阵，所以 $\boldsymbol{U}=\boldsymbol{L}^{\mathrm{T}}$，则有

$$(\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})^{-1}((1-\tilde{\omega})\boldsymbol{D}+\tilde{\omega}\boldsymbol{L}^{\mathrm{T}})\boldsymbol{x}=\lambda\boldsymbol{x}\ .$$

用矩阵 $(\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})$ 左乘上式的两端，再利用 $\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{L}^{\mathrm{T}}$，有

$$(1-\tilde{\omega})\boldsymbol{D}\boldsymbol{x}+\tilde{\omega}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\lambda(\boldsymbol{D}-\tilde{\omega}\boldsymbol{L})\boldsymbol{x}=\lambda(\boldsymbol{D}+\tilde{\omega}(\boldsymbol{A}-\boldsymbol{D}+\boldsymbol{L}^{\mathrm{T}}))\boldsymbol{x}\ ,$$

化简上式，得

$$(1-\tilde{\omega})(1-\lambda)\boldsymbol{D}\boldsymbol{x}+\tilde{\omega}(1-\lambda)\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\lambda\tilde{\omega}\boldsymbol{A}\boldsymbol{x}\ .$$

用 $\boldsymbol{x}$ 共轭转置向量 $\boldsymbol{x}^*$ 左乘上式两端，得

$$(1-\tilde{\omega})(1-\lambda)\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}+\tilde{\omega}(1-\lambda)\boldsymbol{x}^*\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\lambda\tilde{\omega}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\,.$$

由 $\boldsymbol{A}$ 为对称正定阵可得 $\lambda\neq 1$，所以

$$(1-\tilde{\omega})\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}+\tilde{\omega}\boldsymbol{x}^*\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\frac{\lambda}{1-\lambda}\tilde{\omega}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\,.$$

取上式两端的共轭转置，得

$$(1-\tilde{\omega})\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}+\tilde{\omega}\boldsymbol{x}^*\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\frac{\overline{\lambda}}{1-\overline{\lambda}}\tilde{\omega}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\,.$$

将以上两式相加，得

$$2(1-\tilde{\omega})\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}+\tilde{\omega}\boldsymbol{x}^*(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{x}=\left(\frac{\lambda}{1-\lambda}+\frac{\overline{\lambda}}{1-\overline{\lambda}}\right)\tilde{\omega}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\,,$$

化简上式得

$$(2-\tilde{\omega})\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}=\left(1+\frac{\lambda}{1-\lambda}+\frac{\overline{\lambda}}{1-\overline{\lambda}}\right)\tilde{\omega}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}=\frac{1-|\lambda|^2}{|1-\lambda|^2}\tilde{\omega}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}\,.$$

由于 $\boldsymbol{A}$ 和 $\boldsymbol{D}$ 是实对称正定矩阵，且 $\boldsymbol{x}\neq\boldsymbol{0}$，所以 $\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x}>0$，$\boldsymbol{x}^*\boldsymbol{D}\boldsymbol{x}>0$，且 $0<\tilde{\omega}<2$，从而由上式可得 $|\lambda|<1$，故 $\rho(\boldsymbol{B}_{\tilde{\omega}})<1$，因此松弛迭代法收敛.

容易证明：4.5.3 节例子中的系数矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 0.8\\ 0.8 & 1\end{bmatrix}$ 是实对称正定矩阵，根据定理 10，当 $\tilde{\omega}=1.25$ 时，松弛迭代法收敛.

由定理 9 和定理 10 可知，若系数矩阵 $\boldsymbol{A}$ 是实对称正定矩阵，则松弛迭代法收敛的充要条件是 $0<\omega<2$.

4.6 压缩存储迭代法

从上面的三个迭代法公式可以看出，迭代法计算公式有如下两个特点：

（1）迭代过程中解向量的值只和系数矩阵和右端项值有关，且系数矩阵经简化后不再变化.

（2）所有迭代过程都是计算累加和，计算累加和时系数矩阵中的零元素不影响解的值，内存中可不存放也不使用零元素，这正是迭代法能解大型线性方程组的原因.

为了节省篇幅，这一节只介绍压缩存储 Seidel 迭代法.

4.6.1 压缩存储 Seidel 迭代法

这里的压缩存储是指用一维（向量）数组存放二维（矩阵）数组中的除去对角线元素的所有非零元素，为此必须借助辅助数组以表示各元素的行列位置.

1. 系数矩阵元素行列位置

设系数矩阵 $\boldsymbol{A}$ 中每行有 $r_i(<n)$ 个非零元素（去掉对角线元素），用 p_i 表示第 i 行末元素的

位置，则

$$p_i = \sum_{j=1}^{i} r_i = p_{i-1} + r_i ,\quad i = 1,2,\cdots,n \tag{4.6-1}$$

显然，第 i 行非零元素 a_k 中的序号 $k \in [p_{i-1}+1, p_i]$，由于系数矩阵各行非零元素之间往往都间有零元素，所以为了表示元素 a_k 的列，需要增加 l_k 表示第 k 号非零元素的列号.

对于压缩存储平方根法而言，由于带内元素序号是连续的，且第 i 行首元素列号为 $i-m_i+1$，m_i 为 i 行（半）带宽，故可不存放各元素列号.

4.6.2　压缩存储 Seidel 迭代法计算公式

将式（4.2-1）中的 a_{ij} 用一维数组表示，使用

$$\begin{cases} x_i^{(m+1)} = \left(b_i - \sum\limits_{k=r_{i-1}+1}^{k'} a_k x_{l_k}^{(m+1)} + \sum\limits_{k=k'+1}^{r_i} a_k x_{l_k}^{(m)} \right) \Big/ a_{ii} & m = 0,1,\cdots \\ k' = \max\limits_{k} l_k < i & i = 1,2,\cdots,n \\ \left\| \boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)} \right\| > \varepsilon \end{cases} \tag{4.6-2}$$

实际上，Seidel 迭代法计算公式中无论是 $x_{l_k}^{(m+1)}$ 还是 $x_{l_k}^{(m)}$，对于下标 l_k 来讲，都是刚算出的 x_{l_k}，且一旦新的 x_{l_k} 算出后，原值就不再起作用，因而式（4.6-2）可改写成

$$\begin{aligned} x_i &= \left(b_i - \sum_{k=r_{i-1}+1}^{r_i} a_k x_{l_k} \right) \Big/ a_{ii} \\ &= b_i^* - \sum_{k=r_{i-1}+1}^{r_i} a_k^* x_{l_k} \end{aligned} \quad \begin{array}{l} m = 0,1,\cdots \\ i = 1,2,\cdots,n \\ \left\| \boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)} \right\| > \varepsilon \end{array} \tag{4.6-3}$$

由于数组 $\boldsymbol{a}$ 中未存放对角线元素，所以式（4.6-3）中连加时 l_k 自动满足不为 i，于是可令

$$S = \left\| \boldsymbol{x}^{(m+1)} - \boldsymbol{x}^{(m)} \right\|_1 = \sum_{i=1}^{n} \left| x_i' - x_i \right| \tag{4.6-4}$$

其中，x_i' 为刚计算出的解向量的第 i 个分量，x_i 是 x_i'（即 $x_i^{(m+1)}$）算出前的第 m 轮解的第 i 个分量，即 $x_i^{(m)}$.

4.6.3　压缩存储 Seidel 迭代法计算步骤

1．输入量及其意义

N：方程的阶；

M：方程中除对角线元素之外的非零元素个数；

$a_k (k = 1,2,\cdots,M)$：系数矩阵中的非零元素（去掉对角线元素）；

$l_k(k=1,2,\cdots,M)$：系数矩阵中各非零元素的列号；

$m_i(i=1,2,\cdots,N)$：第 i 行去掉 a_{ii} 后的非零元素的个数；

a_{ii}：第 i 行对角线元素，程序中用 x_i 表示，即迭代前 x_i 表示第 i 行对角线元素，正式迭代计算时，表示数组中的初值 $x_i^{(0)}$；

ep：迭代精度.

2．计算步骤

（1）直接给出各行末非零元素以一维形式出现的序号；

（2）计算 b_i^*，$a_{ij}^*=a_k^*$，即初始化迭代矩阵及右端项：

$$\begin{cases} b_i^*=b_i/a_{ii} & i=1,2,\cdots,N \\ a_k^*=a_k^*/a_{ii} & k=r_{i-1}+1,r_{i-1}+2,\cdots,r_i \end{cases} \tag{4.6-5}$$

（3）第 m 轮迭代时（$i=2,3,\cdots,N$），为了叙述方便，用 x_i 表示当前解向量的第 i 个分量. 迭代前 x_i 表示系数中第 i 个对角线元素；

（4）迭代计算（计算 $x_i^{(m+1)}$，$i=1,2,\cdots,n$）：

① $x_i \Rightarrow S_0$

② 利用公式

$$\begin{cases} x_i=b_i^*-\sum\limits_{k=r_{i-1}+1}^{r_i}a_k^*x_{l_k} \\ S=\sum\limits_{i=1}^{n}|x_i'-x_i|=\sum\limits_{i=1}^{n}|x_i'-S_0| \end{cases}, \quad i=2,3,\cdots,N$$

③ 输出

输出 x_i（$i=2,3,\cdots,N$）.

4.6.4 计算实例

【例】用压缩存储迭代法解方程组

$$\begin{aligned}
10x_1+\ x_2+2x_3 &= 18\\
x_1+15x_2+\ x_3+\ 2x_4 &= 36\\
2x_2+8x_3+\ 5x_4 &= 33\\
x_1+\ x_3+10x_4+2x_5 &= 18\\
x_4+6x_5+3x_6 &= 22\\
2x_2+\ x_5+8x_6+\ x_7 &= 31\\
x_6+5x_7-\ x_8 &= 6\\
x_7+8x_8+\ x_9 &= 20\\
2x_8+4x_9 &= 16
\end{aligned}$$

其 Seidel 程序为

```
#include "math.h"
#define  N  9
#define  M  20

void main()
{
    float a[]={0,1,2,1,1,2,2,5,1,1,2,1,3,2,1,1,1,-1,1,1,2};
    float x[]={0,10,15,8,10,6,8,5,8,4};
    float b[]={0,18,36,33,18,22,31,6,20,16};
    float s,s0,ep;
    int r[]={0,2,5,7,10,12,15,17,19,20};
    int l[]={0,2,3,1,3,4,2,4,1,3,5,4,6,2,5,7,6,8,7,9,8};
    int i,k;
    scanf("%f",&ep);
    for(i=1; i<=N; i++)
       b[i]=b[i]/x[i];
    for(i=1; i<=N; i++)
       for(k=r[i-1]+1; k<=r[i]; k++)
          a[k]=a[k]/x[i];
    s=2*ep;
    while(s>ep)
    {
       s=0;
       for(i=1; i<=N; i++)
       {
          s0=x[i];
          for(k=r[i-1]+1; k<=r[i]; k++)
             x[i]=x[i]-a[k]*x[l[k]];
          s=s+fabs(s0-x[i]);
       }
    }
    for(i=1; i<=N; i++)
       printf("x%d=%f\n",i,x[i]);
}
```

用计算机计算出的结果如下：

$$x_1 = 0.999\ 986, x_2 = 1.999\ 999, x_3 = 3.000\ 009, x_4 = 1.000\ 001, x_5 = 2.000\ 000$$
$$x_6 = 3.000\ 000, x_7 = 1.000\ 000, x_8 = 2.000\ 000, x_9 = 3.000\ 000.$$

几点说明如下：

（1）对于大型线性代数方程组求解问题，即所有输入量较大的问题，通常应先将数据写入文件中，经多次校正后，在程序运行时才读入．现场录入大量数据难以保证正确性，采用上面的办法在说明时以初值方式对数组元素赋值，目的是为了教学方便，使程序、算法和数据之间的关系明确．

（2）程序中的数组 $\boldsymbol{x}$ 开始时表示系数矩阵中的对角线元素，其值迭代时作为 $\boldsymbol{x}^{(0)}$；若要提高程序的可读性，可以在说明时于程序中增加数组 *ai*，且将数组 *ai* 和数组 $\boldsymbol{x}$ 说明成共用. 对于大型线性方程组，例如 1000 阶方程组，即使是稀疏矩阵，非零元素的个数也常常要超过 10 000 个. 此时，设计程序时能少用一个数组就尽量少用一个数组. 相反，对于某些不必要考虑内存开销的算法，适当增加数组能增加程序的可读性，会降低程序的设计难度和调试难度.

（3）Gauss-Seidel 迭代法的特点是，总是以当前最新算出的各 x_i 作为计算其他分量的基础，因而程序中可只用一个数组既表示 $\boldsymbol{x}^{(m)}$ 又表示 $\boldsymbol{x}^{(m+1)}$.

（4）因为只使用一个数组表示 $\boldsymbol{x}^{(m)}$ 和 $\boldsymbol{x}^{(m+1)}$，既减少了内存开销，又避免了数据传递. 对于需要考虑效率的问题应尽量考虑效率，若不采用压缩存储，由于内存和时间充裕，所以不必加以考虑.

压缩存储 Seidel 迭代法程序本身的复杂度低，但程序设计难度大，这是计算方法和程序设计中的特例. 方程组的阶超过 5 以后不宜手算，阶数低于 5 时压缩存储又无意义，故不给出手算的实例，也不给出习题.

本章一开头就强调迭代法是求解大中型线性代数方程组的适用方法，实际上只有各种压缩存储迭代法才具有这种功能，一般迭代法比直接法慢且内存开销大（无法利用矩阵的对称性，且要存放 $\boldsymbol{x}^{(m)}$ 和 $\boldsymbol{x}^{(m+1)}$）.

对于超大型方程组，则须用分块算法或并行算法（前面介绍的三种迭代算法，Jacobi 算法易改成并行算法）.

习题 4

1. 取 $\boldsymbol{x}^{(0)}=(0,0,0)^{\mathrm{T}}$，分别用 Jacobi 迭代法和 Gauss-Seidel 迭代法求解下列方程组，要求保留四位有效数字：

（1）$\begin{cases} 10x_1-2x_2-x_3=3 \\ -2x_1+10x_2-x_3=15 \\ -x_1-2x_2+5x_3=10 \end{cases}$　　（2）$\begin{cases} 8x_1-3x_2+2x_3=20 \\ 4x_1+11x_2-x_3=33 \\ 2x_1+x_2+4x_3=12 \end{cases}$

2. 取 $\omega=1.25$，$\boldsymbol{x}^{(0)}=(1,1,1)^{\mathrm{T}}$，用松弛法求解如下方程组，要求精度为 $\frac{1}{2}\times10^{-4}$：

$$\begin{cases} 4x_1+3x_2=24 \\ 3x_1+4x_2-x_3=30 \\ -x_2+4x_3=-24 \end{cases}$$

3. 设线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 的系数矩阵分别为

（1）$\boldsymbol{A}=\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$　　（2）$\boldsymbol{A}=\begin{bmatrix} 2 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & -2 \end{bmatrix}$

试分别讨论 Jacobi 迭代法和 Gauss-Seidel 迭代法的敛散性.

4．已知方程组

$$\begin{cases} x_1 + 0.5x_2 = 0.5 \\ 0.5x_1 + x_2 = -0.5 \end{cases}$$

证明用 Jacobi 迭代法解此方程组是收敛的；若将此方程组中的两个方程交换，请证明交换后的方程组用 Jacobi 迭代法求解是发散的.

5．确定系数矩阵中常数 a 的取值范围，以便使用 Gauss-Seidel 迭代法解 $\boldsymbol{Ax} = \boldsymbol{b}$ 时收敛.

$$\boldsymbol{A} = \begin{bmatrix} 1 & -a \\ -a & 1 \end{bmatrix}$$

6．确定系数矩阵中常数 a 的取值范围，以便使用 Jacobi 迭代法解 $\boldsymbol{Ax} = \boldsymbol{b}$ 时收敛．若 $a = 3$，试判断 Gauss-Seidel 迭代法是否收敛.

$$\boldsymbol{A} = \begin{bmatrix} a & 1 & 3 \\ 1 & a & 2 \\ -3 & 2 & a \end{bmatrix}$$

第 5 章　特征值数值算法

理论上讲，矩阵特征值可通过特征方程求出，但对于高阶矩阵这类算法不仅无法求出理论解，而且难以求出满意的数值解．对于对称矩阵，理论上讲，可通过相似变换把矩阵变成 λ 矩阵从而求出所有特征值，这种变换存在，但未给出构造性算法．特征值及特征向量实际上常用数值算法给出．

5.1　幂法

第 4 章讲到用谱半径和特征值大小来判别迭代法解方程组的收敛性．对于力学，人们更关心其固有频率，这些都和矩阵的最大特征值有关，幂法的功能就是求矩阵的按模最大特征值和对应特征向量．

5.1.1　幂法计算公式

设矩阵 $\boldsymbol{A}$ 有 n 个特征向量 $\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_n$，对应的特征值为 $\lambda_1,\lambda_2,\cdots,\lambda_n$，不妨假设

$$|\lambda_1|\geqslant|\lambda_2|\geqslant\cdots\geqslant|\lambda_n|,$$

显然，任何非零向量 $\boldsymbol{x}\in\boldsymbol{R}^n$ 都可以由 $\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_n$ 的线性组合表示．

设有 $\boldsymbol{x}^{(0)}(\in\boldsymbol{R}^n)\neq\boldsymbol{0}$，$\boldsymbol{x}^{(0)}=\sum_{i=1}^{n}c_i\boldsymbol{u}_i=c_1\boldsymbol{u}_1+c_2\boldsymbol{u}_2+\cdots+c_n\boldsymbol{u}_n$，若 $c_1\neq0$，令

$$\boldsymbol{x}^{(1)}=\boldsymbol{A}\boldsymbol{x}^{(0)}=c_1\lambda_1\boldsymbol{u}_1+c_2\lambda_2\boldsymbol{u}_2+\cdots+c_n\lambda_n\boldsymbol{u}_n,$$

由此有

$$\boldsymbol{x}^{(m)}=\boldsymbol{A}\boldsymbol{x}^{(m-1)}=\boldsymbol{A}^m\boldsymbol{x}^{(0)}=c_1\lambda_1^n\boldsymbol{u}_1+c_2\lambda_2^n\boldsymbol{u}_2+\cdots+c_n\lambda_n^n\boldsymbol{u}_n.$$

当 $|\lambda_1|>|\lambda_2|\geqslant|\lambda_3|\geqslant\cdots\geqslant|\lambda_n|$ 时，有

$$\begin{aligned}\lim_{m\to\infty}\boldsymbol{x}^{(m)}&=\lim_{m\to\infty}\lambda_1^m\left(c_1\boldsymbol{u}_1+c_2\left(\frac{\lambda_2}{\lambda_1}\right)^m\boldsymbol{u}_2+\cdots+c_n\left(\frac{\lambda_n}{\lambda_1}\right)^m\boldsymbol{u}_n\right)\\&=\lim_{m\to\infty}c_1\lambda_1^m\boldsymbol{u}_1\end{aligned}\qquad(5.1\text{-}1)$$

因 $\boldsymbol{x}^{(m)}=\boldsymbol{A}^m\boldsymbol{x}^{(0)}$，上面的算法称为乘幂法，简称幂法．

由于当 $|\lambda_1|<1$ 时，$\lim_{m\to\infty}\lambda_1^m=0$；$\lambda_1>1$ 时，$\lim_{m\to\infty}\lambda_1^m=\infty$，所以由式（5.1-1）可能什么都得不到．计算按模最大特征值及特征向量时，通常不用式（5.1-1）．

5.1.2　实用幂法

实用幂法就是归一化幂法，原因是要避开 $\lim\limits_{m\to\infty}\lambda_1^m=0\,(|\lambda_1|<1)$和 $\lim\limits_{m\to\infty}\lambda_1^m=\infty(|\lambda_1|>1)$．归一化是指将 $\boldsymbol{x}^{(m)}$ 变成 $\infty-$ 范数为 1 的 $\boldsymbol{y}^{(m)}$ 的幂法，常取 $\boldsymbol{x}^{(0)}=\boldsymbol{y}^{(0)}=(1,0,\cdots,0)$ 或 $(1,1,\cdots,1),\cdots$．实用幂法的计算公式为

$$\begin{cases}\boldsymbol{x}^{(m+1)}=\boldsymbol{A}\boldsymbol{y}^{(m)}\\ x_i^{(m+1)}=\sum\limits_{j=1}^{n}a_{ij}y_j^{(m)} & i=1,2,\cdots,n\\ \boldsymbol{y}^{(m+1)}=\boldsymbol{x}^{(m+1)}\Big/\left\|\boldsymbol{x}^{(m+1)}\right\|_\infty & m=0,1,\cdots\\ \varepsilon_1=\sum\limits_{i=1}^{n}|\,y_i^{(m+1)}-y_i^{(m)}\,|>\varepsilon,\ \varepsilon_2=\sum\limits_{i=1}^{n}|\,y_i^{(m+1)}+y_i^{(m)}\,|>\varepsilon\end{cases}\tag{5.1-2}$$

当 $s_1\leqslant\varepsilon$ 时取 $\lambda_1=\alpha_m$，当 $s_2\leqslant\varepsilon$ 时取 $\lambda_1=-\alpha_m$，由式（5.1-1）和式（5.1-2）有

$$\begin{cases}\boldsymbol{x}^{(m+1)}=\boldsymbol{A}\boldsymbol{y}^{(m)}=\dfrac{c_1\lambda_1^{m+1}\boldsymbol{u}_1+c_2\lambda_2^{m+1}\boldsymbol{u}_2+\cdots+c_n\lambda_n^{m+1}\boldsymbol{u}_n}{\alpha_1\alpha_2\cdots\alpha_m}\\ \qquad=\dfrac{c_1\lambda_1^{m+1}}{\alpha_1\alpha_2\cdots\alpha_m}\left(c_1\boldsymbol{u}_1+c_2\left(\dfrac{\lambda_2}{\lambda_1}\right)^{m+1}\boldsymbol{u}_2+\cdots+c_n\left(\dfrac{\lambda_n}{\lambda_1}\right)^{m+1}\boldsymbol{u}_n\right)\\ \boldsymbol{y}^{(m)}=\dfrac{c_1\lambda_1^{m}\boldsymbol{u}_1+c_2\lambda_2^{m}\boldsymbol{u}_2+\cdots+c_n\lambda_n^{m}\boldsymbol{u}_n}{\alpha_1\alpha_2\cdots\alpha_m} & m=0,1,\cdots\\ \qquad=\dfrac{c_1\lambda_1^{m}}{\alpha_1\alpha_2\cdots\alpha_m}\left(c_1\boldsymbol{u}_1+c_2\left(\dfrac{\lambda_2}{\lambda_1}\right)^{m}\boldsymbol{u}_2+\cdots+c_n\left(\dfrac{\lambda_n}{\lambda_1}\right)^{m}\boldsymbol{u}_n\right)\\ \alpha_i=\left\|\boldsymbol{x}^{(i)}\right\|_\infty\end{cases}\tag{5.1-3}$$

当 $|\lambda_1|>|\lambda_2|\geqslant|\lambda_3|\geqslant\cdots\geqslant|\lambda_n|$ 时，有

$$\begin{cases}\lim\limits_{m\to\infty}\boldsymbol{x}^{(m+1)}=\dfrac{c_1\lambda_1^{m+1}}{\alpha_1\alpha_2\cdots\alpha_m}u_1=\lambda\lim\limits_{m\to\infty}\boldsymbol{y}^{(m)}\\ \lim\limits_{m\to\infty}x_i^{(m+1)}=\lambda\lim\limits_{m\to\infty}y_i^{(m)}\\ x_i^{(m+1)}=\sum\limits_{j=1}^{n}a_{ij}y_j^{(m)} & i=1,2,\cdots,n\end{cases}\tag{5.1-4}$$

$$\boldsymbol{y}^{(m+1)}=\boldsymbol{x}^{(m+1)}\Big/\left\|\boldsymbol{x}^{(m+1)}\right\|_\infty=\boldsymbol{x}^{(m+1)}\Big/\alpha_{m+1}$$

5.1.3 实用幂法的计算过程和计算实例

1．计算过程

虽然幂法的功能是求矩阵 $\boldsymbol{A}$ 的按模最大特征值，但人们最感兴趣的是实特征值，故本节不考虑复特征值.

实用幂法的计算过程如下.

（1）选初始向量 $\boldsymbol{x}^{(0)}=(1,0,\cdots,0)^{\mathrm{T}}=\boldsymbol{y}^{(0)}$；

（2）按式（5.1-2）写出具体计算公式；

（3）计算 $S_1=\sum_{i=1}^{n}\left|y_i^{(m+1)}-y_i^{(m)}\right|$ 和 $S_2=\sum_{i=1}^{n}\left|y_i^{(m+1)}+y_i^{(m)}\right|$．当 $S_1\leqslant\varepsilon$ 时，特征向量为 $y^{(m+1)}$，$\lambda=\left\|\boldsymbol{x}^{(m+1)}\right\|_\infty$，计算结束；当 $S_2\leqslant\varepsilon$ 时，特征向量仍为 $y^{(m+1)}$，$\lambda_1=-\left\|\boldsymbol{x}^{(m+1)}\right\|_\infty$，计算结束.

2．计算实例

【例 1】计算 $\boldsymbol{A}=\begin{bmatrix}2&-1&0\\0&2&-1\\0&-1&2\end{bmatrix}$ 的按模最大特征值和对应特征向量，取 $\varepsilon=10^{-3}$.

【解】取 $\boldsymbol{x}^{(0)}=(0,0,1)^{\mathrm{T}}$，$\boldsymbol{y}^{(0)}=\boldsymbol{x}^{(0)}=(0,0,1)^{\mathrm{T}}$．$\boldsymbol{x}^{(1)}=\boldsymbol{A}\boldsymbol{y}^{(0)}=(0,-1,2)^{\mathrm{T}}$，$\alpha_1=2$，$\boldsymbol{y}^{(1)}=(0.-0.5,1)^{\mathrm{T}}$，依次继续迭代，计算结果如表 5.1 所示.

表 5.1 计算结果

i	0	1	2	3	4	5	6	7	8	9
x_i	0	0	0.5	1.2	1.785714	2.195122	2.467272	2.646602	2.765095	2.843652
	0	−1	−2	−2.6	−2.857143	−2.951394	−2.983724	−2.994560	−2.998185	−2.999395
	1	2	2.5	2.8	2.928571	2.975610	2.991862	2.997280	2.999092	2.999697
α_i	1	2	2.5	2.8	2.928571	2.975610	2.991862	2.997280	2.999092	2.999697
y_i	0	0	0.2	0.428571	0.609756	0.737705	0.824661	0.883001	0.921977	0.947980
	0	−0.5	−0.8	−0.928571	−0.975610	−0.991862	−0.997280	−0.999092	−0.999697	−0.999899
	1	1	1	1	1	1	1	1	1	1

从计算结果中可以看出 $\lambda=2.999697$，$\boldsymbol{u}_1=(0.947980,\ -0.999899,\ 1)$.

表中还计算了 $\boldsymbol{y}^{(m+1)}$，实际上当 $\lambda>0$ 时，可直接利用 $\left\|\boldsymbol{y}^{m+1}-\boldsymbol{y}^{m}\right\|\leqslant\varepsilon$ 决定 λ_1，此时 $\lambda_1=\alpha_{m+1}$．本例理论值 $\lambda_1=3$，特征向量理论值为 $(1,\ -1,\ 1)$.

说明：当 $\boldsymbol{x}^{(0)}\doteq\boldsymbol{u}_i$ 或 $\boldsymbol{x}^{(0)}\doteq c_2\boldsymbol{u}_2+\cdots+c_n\boldsymbol{u}_n$ 时，收敛会慢，此时可用另一个 $\boldsymbol{x}^{(0)}=(1,1,\cdots,1)^{\mathrm{T}}$ 代替原 $\boldsymbol{x}^{(0)}$.

5.2　原点平移和逆幂法

由式（5.1-1）和式（5.1-3）可以看出，用幂法或实用幂法计算矩阵 $\boldsymbol{A}$ 的按模最大特征值时，收敛速度取决于 $\left|\dfrac{\lambda_2}{\lambda_1}\right|$，原点平移可改变 $\left|\dfrac{\lambda_2}{\lambda_1}\right|$ 的值.

原点平移就是将原矩阵 $\boldsymbol{A}$ 的特征值都加一个数 p 或减一个数 p .

5.2.1　原点平移算式

【定理】设矩阵 $\boldsymbol{A}$ 的的某一特征值为 λ_i，对应的特征向量为 $\boldsymbol{u}_i$（$i=1,2,\cdots,n$），则 $\boldsymbol{A}-p\boldsymbol{I}$ 的特征值为 λ_i-p，特征向量不变.

【证明】

$$(\boldsymbol{A}-p\boldsymbol{I})\boldsymbol{u}_i=\boldsymbol{A}\boldsymbol{u}_i-p\boldsymbol{I}\boldsymbol{u}_i=\lambda_i\boldsymbol{u}_i-p\boldsymbol{u}_i=(\lambda_i-p)\boldsymbol{u}_i \tag{5.2-1}$$

由定理可以看出，矩阵 $\boldsymbol{A}$ 转换成矩阵 $\boldsymbol{A}-p\boldsymbol{I}$ 实现了原点平移，原点平移时各特征值大小发生了同一变化，但对应的特征向量不变. 由式（5.2-1）还可以看出，由于 λ_1 可以是正数，可以是负数，甚至可以是复数，原点平移后 λ_1 的地位可能会发生变化（不再是按模最大特征值），即使地位不变，$|\lambda_2/\lambda_1|$ 可能变小，也可能变大，因此单一的原点平移意义不大. 原点平移只是将特征值平移，尚需结合特征值算法才能算出特征值。

5.2.2　原点平移加幂法的计算特征值过程和计算实例

1．计算过程

（1）将矩阵做原点平移，得出新矩阵；

（2）写出实用幂法的具体计算公式；

（3）列表写出计算中间结果.

2．计算实例

【例 2】计算 $\boldsymbol{A}=\begin{bmatrix}0.5 & -1 & 0\\ 0 & 0.5 & -1\\ 0 & -1 & 0.5\end{bmatrix}=\begin{bmatrix}2 & -1 & 0\\ 0 & 2 & -1\\ 0 & -1 & 2\end{bmatrix}-1.5\boldsymbol{I}$ 的按模最大特征值，取 $\varepsilon=10^{-3}$.

【解】取 $\boldsymbol{x}^{(0)}=\boldsymbol{y}^{(0)}=(0,0,1)^{\mathrm{T}}$，按公式

$$\begin{cases}\boldsymbol{x}^{(m)}=\boldsymbol{A}\boldsymbol{y}^{(m-1)}\\ \boldsymbol{y}^{(m-1)}=\boldsymbol{x}^{(m-1)}\Big/\left\|\boldsymbol{x}^{(m-1)}\right\|_\infty\end{cases}\qquad \left\|\boldsymbol{y}^{(m)}\pm\boldsymbol{y}^{(m-1)}\right\|_\infty>\varepsilon$$

计算，计算结果如表 5.2 所示.

表 5.2 例 2 的计算结果

m	0	1	2	3	4	5	6	7	8
$x^{(m)}$	0	0	1	1.2	1.428572	1.463415	1.491804	1.498547	1.499945
	0	−1	−1	−1.4	−1.428571	−1.487805	−1.492307	−1.498715	−1.499142
	1	0.5	1.25	1.3	1.464286	1.475610	1.496154	1.497429	1.499571
α_m	1	1	1.25	1.4	1.464286	1.487805	1.496154	1.498715	1.499945
$y^{(m)}$	0	0	0.8	0.857143	0.975610	0.983607	0.997093	0.999889	1
	0	−1	−0.8	−1	−0.975610	−1	−0.997429	−1	−0.999465
	1	0.5	1	0.928571	1	0.992307	1	0.999142	0.999751

计算结果表明 $\lambda_1 = 1.499945$，对应的特征向量为 $(1,\ -0.999465,\ 0.999751)^{\mathrm{T}}$.

矩阵 $\boldsymbol{A} = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ 的最大特征值为 3，平移 1.5 后理论值为 1.5. 本例的收敛速度略高于例 1 的收敛速度. 可以证明，若矩阵 $\boldsymbol{A}$ 的特征值满足 $\lambda_1 > \lambda_2 \geqslant \lambda_3 \geqslant \cdots > \lambda_n > 0$，则平移 $\dfrac{\lambda_2 + \lambda_n}{2}$ 后用幂法计算平移后的特征值收敛最快. 一般情况下，即不知 $\lambda_2, \cdots, \lambda_n$ 或者 λ_i 有负值时，平移多少只能由经验决定，即使像上例中矩阵平移 $\dfrac{\lambda_2 + \lambda_n}{2}$ 后，收敛速度虽然加快了，但并无本质变化. 实际上，原点平移必须配合逆幂法在已知某一特征值的近似值时才有实际意义.

5.2.3 逆幂法

设 $\boldsymbol{A} \in \boldsymbol{R}^{n\times n}$ 是非奇异方阵，有特征值 $\lambda_1, \lambda_2, \cdots, \lambda_n$，与其对应的特征向量为 $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n$，且

$$|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n| > 0,$$

则由 $\boldsymbol{A}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$ 可得 $\boldsymbol{A}^{-1}\boldsymbol{u}_i = \dfrac{1}{\lambda_i}\boldsymbol{u}_i$，即 $\boldsymbol{A}$ 的逆矩阵 $\boldsymbol{A}^{-1}$ 的特征值为 $\dfrac{1}{\lambda_i}$（$i = 1, 2, \cdots, n$），并有

$$\left|\frac{1}{\lambda_n}\right| \geqslant \left|\frac{1}{\lambda_{n-1}}\right| \geqslant \cdots \geqslant \left|\frac{1}{\lambda_1}\right|.$$

由此可知求 $\boldsymbol{A}^{-1}$ 的按模最大特征值及相应的特征向量 $\boldsymbol{u}_n$，就是对 $\boldsymbol{A}$ 求按模最小特征值 λ_n（的倒数）及其相应的特征向量. 这种用 $\boldsymbol{A}^{-1}$ 代替 $\boldsymbol{A}$ 借用幂法求特征值的方法称为逆幂法.

逆幂法的计算公式为

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{A}^{-1}\boldsymbol{x}^{(m)} = (\boldsymbol{A}^{-1})^m \boldsymbol{x}^{(0)} \tag{5.2-2}$$

由于计算 $\boldsymbol{A}^{-1}$ 难免有误差，且计算量较大，不方便，计算时通常用下述办法. 将式（5.2-2）改写成

$$\begin{cases} \boldsymbol{A}\boldsymbol{x}^{(m+1)} = \boldsymbol{x}^{(m)} \\ \boldsymbol{x}^{(0)} = (0, 0, \cdots, 1)^{\mathrm{T}} \end{cases}$$

用解方程的方法解出 $\boldsymbol{x}^{(m+1)}$，当 $\boldsymbol{A}$ 的阶不高时，手算可用 $\boldsymbol{LU}$ 分解法求 $\boldsymbol{x}^{(m+1)}$．也可通过解 n 个方程组 $\boldsymbol{A}\boldsymbol{y}^{(i)}=\boldsymbol{e}_i\,(i=1,2,\cdots,n)$，$\boldsymbol{e}_i$ 是第 i 个分量为 1 的单位向量，求 $\boldsymbol{A}^{-1}=(\boldsymbol{y}^{(1)}\boldsymbol{y}^{(2)}\cdots\boldsymbol{y}^{(n)})$．

逆幂法的具体计算过程如下．

（1）将 $\boldsymbol{A}$ 分解成 $\boldsymbol{LU}$，即 $\boldsymbol{A}=\boldsymbol{LU}$；

（2）解方程组．

令

$$\boldsymbol{LUx}^{(k+1)}=\boldsymbol{Lz}^{(k+1)}=\boldsymbol{x}^{(k)}$$

$$z_1^{(k+1)}=x_1^{(k)}$$

$$z_i^{(k+1)}=x_i^{(k)}-\sum_{j=1}^{i-1}l_{ij}z_j^{(k+1)},\quad i=2,3,\cdots,n$$

解 $\boldsymbol{Ux}^{(k+1)}=\boldsymbol{z}^{(k+1)}$

$$x_n^{(k+1)}=z_n^{(k+1)}\Big/u_{nn}$$

$$x_i^{(k+1)}=\left(z_i^{(k+1)}-\sum_{j=i+1}^{n}u_{ik}x_j^{(k+1)}\right)\Big/u_{ii},\quad i=n-1,n-2,\cdots,1$$

$$\alpha_{k+1}=\max_i\left|x_i^{(k+1)}\right|$$

$$\boldsymbol{y}^{(k+1)}=\boldsymbol{x}^{(k+1)}\Big/\alpha_{k+1}$$

$$\left|\alpha_{k+1}-\alpha_k\right|>\varepsilon$$

当 $\left|\alpha_{k+1}-\alpha_k\right|\leqslant\varepsilon$ 时计算结束．

$\boldsymbol{A}$ 的经原点平移后按模最小特征值的倒数为

$$\lambda_1=\pm\alpha_{k+1}\begin{cases}y_i^{(k+1)}\doteq y_i^{(k)}\\ y_i^{(k+1)}\doteq -y_i^{(k)}\end{cases}$$

$\boldsymbol{A}$ 的对应特征值为

$$\lambda^*=\lambda_0+1/\alpha_{k+1}$$

其中，λ_0 为原点平移量，该特征值对应的特征向量为 $\boldsymbol{y}^{(k+1)}$．

使用 $\boldsymbol{LU}$ 法的原因是，将 $\boldsymbol{A}$ 分解成 $\boldsymbol{LU}$ 后可多次使用，实际上用 Gauss 列主元素法也许会更好．$\boldsymbol{A}$ 转换成 $\boldsymbol{U}$ 的结果可保留，但要存放 $a_{ii}^{(i-1)}$ 和行交换序号．算法相对复杂，为手算方便，书中只介绍 $\boldsymbol{LU}$ 法．

5.2.4　逆幂法计算实例

【例 3】用逆幂法及原点平移计算矩阵 $\boldsymbol{A}=\begin{bmatrix}2&-1&0\\0&2&-1\\0&-1&2\end{bmatrix}$ 接近 3 的特征值及特征向量，取 $p=2.93$，$\varepsilon=10^{-3}$．

【解】对矩阵 $\boldsymbol{A}-2.93\boldsymbol{I}$ 做 $\boldsymbol{LU}$ 分解：

$$\boldsymbol{A}-2.93\boldsymbol{I}=\begin{bmatrix}-0.93 & -1 & 0\\ 0 & -0.93 & -1\\ 0 & -1 & -0.93\end{bmatrix}$$

$$\doteq\begin{bmatrix}1 & & \\ 0 & 1 & \\ 0 & 1.075269 & 1\end{bmatrix}\begin{bmatrix}-0.93 & -1 & 0\\ & -0.93 & -1\\ & & 0.145269\end{bmatrix}$$

$$=\boldsymbol{LU}$$

取 $\boldsymbol{x}^{(0)}=(0,0,1)^{\mathrm{T}}=\boldsymbol{y}^{(0)}$，有

$$\boldsymbol{LU}\boldsymbol{x}^{(1)}=\boldsymbol{x}^{(0)}=\boldsymbol{y}^{(0)},\quad \boldsymbol{y}^{(1)}=\boldsymbol{y}^{(1)}\Big/\left\|\boldsymbol{x}^{(1)}\right\|_{\infty}$$

$$\boldsymbol{LU}\boldsymbol{x}^{(2)}=\boldsymbol{y}^{(1)}$$

$$\cdots$$

计算结果如表 5.3 所示.

表 5.3 例 3 的计算结果

m	0	1	2	3	4
$x^{(m)}$	0	7.959059	12.692311	14.278433	14.266834
	0	−7.401925	−12.803849	−14.267627	−14.268161
	1	6.883781	12.837580	14.266266	14.268633
α_m	1	7.959059	12.837580	14.278433	14.268633
$y^{(m)}$	0	1	0.988684	1	0.999874
	0	−0.930000	−0.997373	−0.999243	0.999967
	1	0.864899	1	0.999148	1
z^m		0	0.988684	1	1
		0	−0.997373	−0.999243	−0.999243
		1	2.072444	2.073603	2.072790

由表 5.3 可以看出，$\boldsymbol{A}-2.93\boldsymbol{I}$ 按模最小特征值为 $1/\alpha_4$，而接近 2.93 的特征值为 $2.93+1/\alpha_4=2.93+0.0700838=3.0000838$．对应的特征向量为 $(0.999874,\ 0.999967,\ 1)^{\mathrm{T}}$．

5.3 实对称矩阵特征值数值算法——对分法

对分法只能计算（实）三对角对称矩阵的各个特征值．因实对称矩阵可通过镜面反射变换转换成三对角矩阵，因而对分法只是（实）对称矩阵特征值算法．

5.3.1　镜面反射矩阵及其性质

【定义 1】称 $\boldsymbol{H}=\boldsymbol{I}-2\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}$，$\|\boldsymbol{u}\|_2=1$ 为镜面反射矩阵.

镜面反射矩阵也称为 Householder 矩阵．显然，

$$\boldsymbol{H}=\boldsymbol{I}-2\frac{\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}}{\|\boldsymbol{v}\|_2^2},\quad \boldsymbol{v}\neq\boldsymbol{0}$$

也是镜面反射矩阵.

镜面反射矩阵具有下面的基本性质：

（1）对称性：$\boldsymbol{H}=\boldsymbol{H}^{\mathrm{T}}$；

（2）正交性：$\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{I}$；

（3）对合性：$\boldsymbol{H}\boldsymbol{H}=\boldsymbol{I}$.

证明略.

【定理 1】对于任意 $\boldsymbol{x}=\boldsymbol{v}+\boldsymbol{w}=c\boldsymbol{u}+\boldsymbol{w}$，$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{u}=\boldsymbol{0}$，有

$$\boldsymbol{H}\boldsymbol{x}=-\boldsymbol{v}+\boldsymbol{w}.$$

【证明】

$$\begin{aligned}\boldsymbol{H}\boldsymbol{x}&=(\boldsymbol{I}-2\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}})(\boldsymbol{v}+\boldsymbol{w})\\&=\boldsymbol{v}+\boldsymbol{w}-2\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}(\boldsymbol{v}+\boldsymbol{w})\\&=\boldsymbol{v}+\boldsymbol{w}-2\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{v}-2\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{w}\\&=\boldsymbol{v}+\boldsymbol{w}-2c\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}\\&=\boldsymbol{v}+\boldsymbol{w}-2\boldsymbol{v}\\&=-\boldsymbol{v}+\boldsymbol{w}\end{aligned}$$

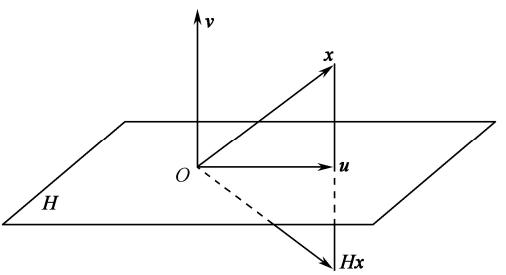

图 5.1　镜面反射图

这正是称 $\boldsymbol{H}$ 矩阵为镜面反射矩阵而称 $\boldsymbol{x}\to\boldsymbol{H}\boldsymbol{x}$ 变换为镜面反射变换的原因（见图 5.1）.

【定理 2】设 $\boldsymbol{b},\boldsymbol{v}\in\boldsymbol{R}^n$，$\boldsymbol{b}\neq\boldsymbol{v}$，$\|\boldsymbol{b}\|_2=\|\boldsymbol{v}\|_2$，则存在 $\boldsymbol{H}$，使得 $\boldsymbol{H}\boldsymbol{b}=\boldsymbol{v}$.

【证明】取 $\boldsymbol{u}=\boldsymbol{b}-\boldsymbol{v}$，作 $\boldsymbol{H}=\boldsymbol{I}-2\dfrac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{(\boldsymbol{b}-\boldsymbol{v})^{T}(\boldsymbol{b}-\boldsymbol{v})}$.

$$\begin{aligned}\boldsymbol{H}\boldsymbol{b}&=\left(\boldsymbol{I}-2\frac{\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}}{(\boldsymbol{b}-\boldsymbol{v})^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{v})}\right)\boldsymbol{b}\\&=\boldsymbol{b}-2\frac{(\boldsymbol{b}-\boldsymbol{v})(\boldsymbol{b}-\boldsymbol{v})^{\mathrm{T}}\boldsymbol{b}}{(\boldsymbol{b}-\boldsymbol{v})^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{v})}\\&=\boldsymbol{b}-2\frac{(\boldsymbol{b}-\boldsymbol{v})^{\mathrm{T}}\boldsymbol{b}}{(\boldsymbol{b}-\boldsymbol{v})^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{v})}(\boldsymbol{b}-\boldsymbol{v})\\&=\boldsymbol{b}-\frac{2\boldsymbol{b}^{\mathrm{T}}\boldsymbol{b}-2\boldsymbol{v}^{\mathrm{T}}\boldsymbol{b}}{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{b}-\boldsymbol{v}^{\mathrm{T}}\boldsymbol{b}-\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}}(\boldsymbol{b}-\boldsymbol{v})\\&=\boldsymbol{b}-(\boldsymbol{b}-\boldsymbol{v})\\&=\boldsymbol{v}\end{aligned}$$

上式中 $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{b}=\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}$，$\boldsymbol{v}^{\mathrm{T}}\boldsymbol{b}=\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}$.

上面的定理说明，两个长度相等的向量，可通过一个镜面反射变换，使一个向量为另一个向量的像.

【定理 3】 设 $\boldsymbol{G}_{n-r}$ 为 $n-r$ 阶镜面反射矩阵，则

$$\boldsymbol{H}=\begin{bmatrix}\boldsymbol{I}_r & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{G}_{n-r}\end{bmatrix}$$

是镜面反射矩阵.

【证明】 设 $\boldsymbol{G}=\boldsymbol{I}_{n-r}-2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}$，$\|\boldsymbol{v}\|_2=\|\boldsymbol{v}\|_2^2=1$.

取 $\boldsymbol{u}=(\boldsymbol{0}_r^{\mathrm{T}},\boldsymbol{v}^{\mathrm{T}})$，$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}=\boldsymbol{0}_r^{\mathrm{T}}+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}=1$.

$$\begin{aligned}\boldsymbol{I}_n-2\boldsymbol{u}\boldsymbol{u}^{\mathrm{T}}&=\boldsymbol{I}_n-2\begin{bmatrix}\boldsymbol{0}_r\\ \boldsymbol{v}\end{bmatrix}[\boldsymbol{0}_r^{\mathrm{T}},\boldsymbol{v}^{\mathrm{T}}]\\ &=\begin{bmatrix}\boldsymbol{I}_r & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}_{n-r}-2\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}\end{bmatrix}\\ &=\boldsymbol{H}\end{aligned}$$

5.3.2 实对称矩阵三对角化

这里先给出一个实例，再给出通用计算公式和通用计算过程.

【例 1】 用 Householder 变换将下面的矩阵三对角化：

$$\boldsymbol{A}_1=\boldsymbol{A}=\begin{bmatrix}6&2&3&1\\2&5&4&8\\3&4&9&1\\1&8&1&7\end{bmatrix}$$

【解】 取 $\boldsymbol{b}_1=(6,2,3,1)^{\mathrm{T}}$，$\boldsymbol{v}_1=(6,-\sqrt{2^2+3^2+1^2},0,0)^{\mathrm{T}}$，$\boldsymbol{u}_1=\boldsymbol{b}-\boldsymbol{v}_1=(0,2+\sqrt{14},3,1)^{\mathrm{T}}$.

$$\boldsymbol{H}_1=\boldsymbol{I}_4-\frac{2\boldsymbol{u}_1\boldsymbol{u}^{\mathrm{T}}}{\boldsymbol{u}_1^{\mathrm{T}}\boldsymbol{u}_1}=\begin{bmatrix}1&0&0&0\\0&-0.5345&-0.8018&-0.2673\\0&-0.8018&-0.5811&-0.1396\\0&-0.2673&-0.1396&0.9535\end{bmatrix}$$

$$\boldsymbol{A}_2=\boldsymbol{H}_1\boldsymbol{A}\boldsymbol{H}_1=\begin{bmatrix}6&-3.7417&0&0\\-3.7417&13.8756&1.8079&-3.1394\\0&1.8079&4.2912&-6.0079\\0&-3.1394&-6.0079&2.8512\end{bmatrix}$$

取

$$\boldsymbol{b}_2=(-3.7417,13.8756,1.8079,-3.1384)^{\mathrm{T}}$$

$$\boldsymbol{v}_2=(-3.7417,13.8756,-\sqrt{1.8079^2+3.1394^2},0)^{\mathrm{T}}$$

$$\boldsymbol{u}_2=\boldsymbol{b}_2-\boldsymbol{v}_2=(0,0,5.4307,-3.1394)^{\mathrm{T}}$$

$$\boldsymbol{H}_2=\boldsymbol{I}_4-\frac{2\boldsymbol{u}_2\boldsymbol{u}_2^{\mathrm{T}}}{\boldsymbol{u}_2^{\mathrm{T}}\boldsymbol{u}_2}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&-0.4990&0.8666\\0&0&0.8666&0.9990\end{bmatrix}$$

$$\boldsymbol{A}_3=\boldsymbol{H}_2\boldsymbol{A}_2\boldsymbol{H}_2=\begin{bmatrix}6&-3.7417&0&0\\-3.7417&13.8576&-3.6227&0\\0&-3.6227&8.4058&-3.6387\\0&0&-3.6387&-1.2634\end{bmatrix}$$

$\boldsymbol{A}_3$ 已是三对角矩阵.

5.3.3　实对称矩阵三对角化算法

实例中四阶对称矩阵三对角化需要做二次 $\boldsymbol{AHA}$ 计算．下面是三对角化的通用公式和通用计算过程.

设 $\boldsymbol{A}_1=\boldsymbol{A}$，$\boldsymbol{A}_2=\boldsymbol{H}_1\boldsymbol{A}_1\boldsymbol{H}_1$，$\boldsymbol{A}_{i+1}=\boldsymbol{H}_i\boldsymbol{A}_i\boldsymbol{H}_i$，令

$$\boldsymbol{A}_i=\{a_{jk}^{(i)}\}$$

$\boldsymbol{H}_i$ 和 $\boldsymbol{A}_{i+1}$ 的计算公式和计算过程如下.

1．计算 $\boldsymbol{H}_i$

（1）取 $\boldsymbol{b}^{(i)}=\left(a_{i1}^{(i)},a_{i2}^{(i)},\cdots,a_{in}^{(i)}\right)^{\mathrm{T}}$

$$\boldsymbol{v}^{(i)}=\left(b_1,b_1,\cdots,b_i,-\mathrm{sgn}(b_{i+1})\left(\sum_{j=i+1}^{n}b_j^2\right)^{1/2},0,\cdots,0\right)^{\mathrm{T}}$$

$$\boldsymbol{u}^{(i)}=\boldsymbol{b}^{(i)}-\boldsymbol{v}^{(i)}$$

$$\boldsymbol{H}_i=\boldsymbol{I}-2\frac{\boldsymbol{u}^{(i)}(\boldsymbol{u}^{(i)})^{\mathrm{T}}}{(\boldsymbol{u}^{(i)})^{\mathrm{T}}\boldsymbol{u}^{(i)}}$$

式中 $\mathrm{sgn}(b_{i+1})$ 表示 b_{i+1} 的数符，目的在于避免两个相近数相减.

（2）计算 $\boldsymbol{A}_{i+1}$.

以上计算的带过程和范围的计算公式为

令 $\boldsymbol{H}_i=\{h_{jk}^{(i)}\}$，$S=(\boldsymbol{u}^{(i)})^{\mathrm{T}}\boldsymbol{u}^{(i)}=\sum_{j=i}^{n}(u_j^{(i)})^2$

$$h_{jk}=\begin{cases}-2u_j^{(i)}u_k^{(i)}/S & j\neq k\quad i=1,2,\cdots,n-2\\1-2u_j^{(i)}u_k^{(i)}/S & j=k\quad k=1,2,\cdots,n\end{cases}\tag{5.3-1}$$

2．计算 $\boldsymbol{A}_{i+1}=\boldsymbol{H}_i\boldsymbol{A}_i\boldsymbol{H}_i$

设 $\boldsymbol{A}1_{i+1}=\boldsymbol{H}_i\boldsymbol{A}_i$，$\boldsymbol{A}_{i+1}=\boldsymbol{A}1_{i+1}\boldsymbol{H}_i$

$$\begin{cases} a1_{jk}^{(i+1)}=\sum_{l=1}^{n}h_{jl}^{(i)}a_{lk}^{(i)} \\ a_{jk}^{(i+1)}=\sum_{l=1}^{n}a1_{jl}^{(i)}h_{lk}^{(i)} \end{cases} \quad j,k=1,2,\cdots,n \tag{5.3-2}$$

5.3.4 实对称矩阵三对角化程序

下面是对称矩阵三对角化的通用程序.

```
#include "math.h"
#define  N  4

void main()
{
    int i,j,k,l;
    float s,sn;
    float A[N+1][N+1],A1[N+1][N+1],H[N+1][N+1],b[N+1],v[N+1]={0},u[N+1];
    for(i=1; i<=N; i++)
        for(j=1; j<=N; j++)
            scanf("%f",&A[i][j]);
    for(i=1; i<=N-2; i++)
    {
        for(j=1; j<=N; j++)
        {
            b[j]=A[i][j];
            printf("%13.6f",b[j]);
        }
        printf("\n");
        for(j=1; j<=i; j++)
            v[j]=b[j];
        s=0.0;
        for(j=i+1; j<=N; j++)
            s=s+b[j]*b[j];
        if( b[i+1]>0)
            sn=-1.0;
        else
            sn=1;
        v[i+1]=sn*sqrt(s);
        for(j=1; j<=N; j++)
            printf("%13.6f",v[j]);
        printf("\n");
```

```
    for(j=1; j<=N; j++)
    {
       u[j]=b[j]-v[j];
       printf("%13.6f",u[j])
    }
    printf("\n");
    s=0;
    for(j=1; j<=N; j++)
       s=s+u[j]*u[j];
    for(j=1; j<=N; j++)
       for(k=1; k<=N; k++)
       {
          if(j!=k)
             H[j][k]=-2*u[j]*u[k]/s;
          else
             H[j][k]=1.0-2*u[j]*u[k]/s;
       }
    for(j=1; j<=N; j++)
    {
       for(k=1; k<=N; k++)
          printf("%13.6f",H[j][k]);
       printf("\n");
    }
    for(j=1; j<=N; j++)
       for(k=1; k<=N; k++)
       {
          s=0.0;
          for(l=1; l<=N; l++)
             s=s+H[j][l]*A[l][k];
          A1[j][k]=s;
       }
    for(j=1; j<=N; j++)
       for(k=1; k<=N; k++)
       {
          s=0.0;
          for(l=1; l<=N; l++)
             s=s+A1[j][l]*H[l][k];
          A[j][k]=s;
       }
}
printf("=====Results:\n");
for(j=1; j<=N; j++)
{
    for(k=1; k<=N; k++)
       printf("%13.6f", A[j][k]);
```

```
            printf("\n");
        }
    }
```

对于例 1，利用本程序的计算结果为：

```
 6.000000   -3.741657    0.000000   -0.000000
-3.741657   13.857143   -3.622492   -0.000000
 0.000000   -3.622492    8.407243   -3.636914
-0.000000   -0.000000   -3.636914   -1.264386
```

程序说明：

（1）本程序未考虑计算量. 实际上，做 Householder 变换时，只须计算定理 3 中的 $\boldsymbol{G}_n - r$，不必计算 $\boldsymbol{H}$，计算 $\boldsymbol{b}$ 时也只须计算 $\boldsymbol{b}$ 中的 $b_{i+1}, b_{i+2}, \cdots, b_n$. $\boldsymbol{v}$ 只须计算 v_{i+1}. 由于做 $\boldsymbol{H}_i \boldsymbol{A}_i \boldsymbol{H}_i$ 计算时，$\boldsymbol{A}_i$ 中的 i 阶主子矩阵已不会变化，程序也可不考虑这部分计算. 程序中给 $\boldsymbol{v}$ 数组赋初值为 0，目的仅在于减少程序中的语句.

（2）若 $\boldsymbol{A}$ 是稀疏矩阵，先做行列交换使 $\boldsymbol{A}$ 的后半带宽中末列号按升序排列（仍是实对称矩阵），也可仿 $\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ 压缩储存，因计算公式较为复杂，教材中未曾介绍. Householder 变换须和对分法、原点平移、逆幂法配合才是有意义的算法，由于算法所对应的程序图中线性无关环较多，故书中所给程序. 不考虑可读性并给出了中间计算结果，而不给出和压缩存储程序.

5.3.5 求实对称矩阵特征值的对分法

首先讨论三对角对称矩阵的情形.

1. 实对称三对角矩阵的 Sturm 序列

设实对称三对角矩阵

$$\boldsymbol{C} = \begin{bmatrix} c_1 & b_1 & & & & \\ b_1 & c_2 & b_2 & & & \\ & b_2 & c_3 & b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & b_{n-2} & c_{n-1} & b_{n-1} \\ & & & & b_{n-1} & c_n \end{bmatrix}$$

其中 $b_i \neq 0$（$i = 1, 2, \cdots, n$），用 $p_i(\lambda)$ 表示 $\boldsymbol{C} - \lambda \boldsymbol{I}$ 的 i 阶主子行列式，并规定 $p_0(\lambda) = 1$，$b_0 = 0$，则 $p_n(\lambda) = \det(\boldsymbol{C} - \boldsymbol{I}\lambda)$ 的特征多项式可由下式算出：

$$\begin{cases} p_0(\lambda) = 1 \\ p_1(\lambda) = c_1 - \lambda \\ p_2(\lambda) = (c_2 - \lambda) p_1(\lambda) - b_1^2 p_0(\lambda) \\ \qquad \vdots \\ p_i(\lambda) = (c_i - \lambda) p_{i-1}(\lambda) - b_{i-1}^2 p_{i-2}(\lambda) \qquad i = 2, 3, \cdots, n \end{cases} \tag{5.3-3}$$

我们称多项式序列 $\{p_i(\lambda)\mid i=0,1,\cdots,n\}$ 为矩阵 $\boldsymbol{C}$ 的 Sturm 序列.

下面是 Sturm 序列的性质.

【性质 1】 $p_i(\lambda)=0$（$i=1,2,\cdots,n$）仅有实根.

【证明】因 $p_i(\lambda)$ 所对应矩阵是实对称矩阵，所以 $p_i(\lambda)=0$ 只有实根.

【性质 2】相邻的两个多项式 $p_i(\lambda)$ 和 $p_{i-1}(\lambda)$ 无公共零点.

【证明】用反证法证明这一性质.

设 $p_i(\lambda)=p_{i-1}(\lambda_0)=0$，由定义 $p_i(\lambda_0)=(c_i-\lambda_0)p_{i-1}(\lambda_0)-b_{i-1}^2p_{i-2}(\lambda_0)$ 可知 $p_{i-2}(\lambda_0)=0$，继续推导下去则有 $p_0(\lambda_0)=0$，这与定义矛盾，所以性质成立.

【性质 3】设 $p_i(\lambda_0)=0$，则 $p_{i-1}(\lambda_0)p_{i+1}(\lambda_0)<0$.

【证明】由性质 2 有 $p_{i-1}(\lambda_0)\neq 0$，由定义有

$$p_{i+1}(\lambda_0)=(c_i-\lambda_0)p_i(\lambda_0)-b_{i-1}^2p_{i-1}(\lambda_0)=-b_{i-1}^2p_{i-1}(\lambda_0)\neq 0$$

$$\frac{p_{i+1}(\lambda_0)}{p_{i-1}(\lambda_0)}=-b_{i-1}^2<0$$

$$p_{i-1}(\lambda_0)p_{i+1}(\lambda_0)<0.$$

【性质 4】 $p_i(\lambda)=0$ 只有单根，且把 $p_{i+1}(\lambda)=0$（$i=1,2,\cdots,n-1$）的根严格地隔离开来.

【证明】 $p_i(\lambda)$ 的最高次项为 $(-\lambda)^i$，故在 $\lambda=-\infty$ 的邻域内，$p_i(\lambda)>0$，记为 $p_i(-)>0$；而在 $\lambda=\infty$ 的邻域内，都和 $(-\lambda)^i$ 同号，记为 $\mathrm{sign}p_i(\infty)$.

下面用归纳法来证明该性质.

当 $i=1$ 时，$p_1(\lambda)=c_1-\lambda$，c_1 是 $p_1(\lambda)=0$ 的根，另一方面，$p_2(\lambda)=-b_1^2<0$，$p_2(-\infty)>0$，$p_2(\infty)>0$，即 $p_2(\lambda)$ 在区间 $(-\infty,c_1)$ 和区间 $(c_1,+\infty)$ 的端点异号，所以 $p_2(\lambda)$ 在区间 $(-\infty,c_1)$ 和区间 $(c_1,+\infty)$ 内各有一个根，也就是说 $i=1$ 时，性质成立.

假设 $i=k-1$ 时性质成立，即 $p_{k-1}(\lambda)=0$ 和 $p_k(\lambda)=0$ 全是单根，且 $p_{k-1}(\lambda)=0$ 的根把 $p_k(\lambda)=0$ 的根全部隔离开来. 设 $p_{k-1}(\lambda)=0$ 的根由小到大的顺序为

$$x_1<x_2<\cdots<x_{k-1}$$

$p_k(\lambda)=0$ 的根的顺序为

$$y_1<y_2<\cdots<y_k$$

则有

$$y_1<x_1<y_2<x_2<y_3<\cdots<y_{k-1}<x_{k-1}<y_k$$

当 $i=k$ 时，由于 $p_{k-1}(-\infty)>0$，$p_{k-1}(x_1)=0$，所以

$$p_{k-1}(y_1)>0,\quad p_{k-1}(y_2)<0,\quad p_{k-1}(y_3)>0,\quad \cdots$$

$p_{k-1}(y_j)$ 的符号为 $(-1)^{j+1}$，由性质 3 有 $p_{k+1}(y_j)$ 的符号为 $(-1)^j$，即

$$p_{k+1}(-\infty)>0,\quad p_{k+1}(y_1)<0,\quad p_{k+1}(y_2)>0,\quad \cdots$$

于是，在 $k+1$ 个区间 $(-\infty,y_1),(y_1,y_2),\cdots,(y_k,+\infty)$ 内都有 $p_k(\lambda)=0$ 的根，而 $p_{k+1}(\lambda)=0$ 只有 $k+1$ 个根，所以每个区间只有一个根. 证毕.

【性质 5】设$1 \leqslant i \leqslant n-1$，且$\lambda_0$是$p_i(\lambda)=0$的根，则当$\varepsilon(>0)$充分小时，$p_{i-1}(\lambda)$和$p_i(\lambda)$在区间$(\lambda_0-\varepsilon,\lambda_0)$内同号；$p_i(\lambda)$和$p_{i+1}(\lambda)$在区间$(\lambda_0,\lambda_0+\varepsilon)$内同号.

【证明】设λ_0是$p_i(\lambda)=0$的第k个根，并用$\lambda_{i,k}$表示，由性质 4 有$\lambda_{i-1,k-1}<\lambda_{i,k}$，且在区间$(\lambda_{i-1,k-1},\lambda_0)$内没有$p_{i-1}(\lambda)=0$的根，又在区间$(\lambda_{i-1,k-1},\lambda_{k,i})$内，$p_{i-1}(\lambda)$和$p_i(\lambda)$的符号都是$(-1)^{k-1}$，所以$p_{i-1}(\lambda)$和$p_i(\lambda)$在区间$(\lambda_0-\varepsilon,\lambda_0)$内同号. 类似地，可以证明$p_i(\lambda)$和$p_{i+1}(\lambda)$在区间$(\lambda_0,\lambda_0+\varepsilon)$内同号.

使用对分法前还须了解如下事项：

（1）Sturm 序列$\{p_0(\lambda),p_1(\lambda),\cdots,p_n(\lambda)\}$在某点的连号数.

Sturm 序列在$\lambda=\lambda_0$处为一数列$\{p_0(\lambda_0),p_1(\lambda_0),\cdots,p_n(\lambda_0)\}$，它的连号数由下列规则确定：

① 若$p_{i-1}(\lambda_0)=0\ (i=1,2,\cdots,n)$，则认为$p_{i-1}(\lambda_0)$为正号；

② 若$p_{i-1}(\lambda_0)$与$p_i(\lambda_0)$同号，则$p_{i-1}(\lambda_0)$和$p_i(\lambda_0)$连号，若符号相反则不连号；

③ $\varphi(\lambda_0)=\{p_0(\lambda_0),p_1(\lambda_0),\cdots,p_n(\lambda_0)\}$的连号个数.

例如，对某个 Sturm 序列，当$\lambda_0=4$，$\lambda_0=6$时分别为

$\varphi(4)=\{1,2,3,5,6\}$的连号个数为 4.

$\varphi(6)=\{1,0,-2,3,4\}$的连号个数为 2.

（2）圆盘定理.

圆盘定理是估计矩阵特征值范围的基本定理，它适用于一般矩阵，当然也可用于三对角实对称矩阵.

【定义】设$\boldsymbol{A}=(a_{ij})$是n阶方阵，令

$$\boldsymbol{G}_i=\{z\,|\,|z-a_{ii}|\leqslant d_i\},\quad i=1,2,\cdots,n$$

其中$d_i=\sum_{j=1}^{i-1}|a_{ij}|+\sum_{j=i+1}^{n}|a_{ij}|$，即$d_i$为$\boldsymbol{A}$中第$i$行除了对角线元素之外，其余元素的绝对值之和，称$\boldsymbol{G}_i$为第$i$个圆盘.

【定理 1（圆盘定理）】设$\boldsymbol{A}=(a_{ij})$是n阶方阵，令$\boldsymbol{G}=\boldsymbol{G}_1\cup\boldsymbol{G}_2\cup\cdots\cup\boldsymbol{G}_n$，则$\boldsymbol{A}$的全部特征值都在$\boldsymbol{G}$内.

【证明】当$z\notin\boldsymbol{G}$时，则对$i=1,2,\cdots,n$，$z\notin\boldsymbol{G}$，即$|z-a_{ii}|>d_i$（$i=1,2,\cdots,n$），所以矩阵$\boldsymbol{A}-z\boldsymbol{I}$是严格对角占优矩阵，是非奇异矩阵，即$\det(\boldsymbol{A}-z\boldsymbol{I})\neq 0$，故$z$不是$\boldsymbol{A}$的特征值，也就是说$\boldsymbol{A}$的全部特征值都属于$\boldsymbol{G}$.

由圆盘定理可立即得到如下推论.

【推论】实对称三对角矩阵$\boldsymbol{C}$的特征值满足

$$\begin{cases} m=\min\limits_i(c_i-|b_i|-|b_{i-1}|) \\ M=\max\limits_i(c_i+|b_i|+|b_{i-1}|) \end{cases},$$

它们都属于$[m,M]$.

【证明】实对称三对角矩阵$\boldsymbol{C}$的第i个圆盘为

$$\boldsymbol{G}_i=\{z\,|\,|z-c_j|\leqslant|b_{j-1}|+|b_j|\}\,,\quad i=1,2,\cdots,n$$

设 λ_i 是 $\boldsymbol{C}$ 的任意一个特征值，则由圆盘定理必存在圆盘 $\boldsymbol{G}_i$ 使得 $\lambda_i\in\boldsymbol{G}_i$，于是

$$|\lambda_i-c_j|\leqslant|b_j|+|b_{j-1}|$$

从而

$$m\leqslant c_j-|b_j|-|b_{j-1}|\leqslant c_j+|b_j|+|b_{j-1}|\leqslant M$$

故 $\boldsymbol{C}$ 的所有特征值都属于 $[m,M]$.

现介绍求实对称矩阵特征值的对分法.

【定理 2】矩阵 $\boldsymbol{C}$ 在区间 $[\lambda_0,+\infty)$ 内特征值的个数等于 $\varphi(\lambda_0)$.

【证明】在 $-\infty$ 邻域，$p_i(\lambda)$ 都大于 0，连号数为 n，当 λ 值由 $-\infty$ 的邻域右移时，由于 $p_n(\lambda)$ 的最小零点不小于 $p_i(\lambda)$（$i=1,2,\cdots,n-1$）的零点，一旦 λ 值大于 $p_n(\lambda)$ 的最小零点，则此时 $p_n(\lambda)$ 变号，由于 $p_i(\lambda)$（$i=1,2,\cdots,n$）无共同零点，所以 $p_i(\lambda)\,(i=1,2,\cdots,n-1)$都不变号，这样 $p_n(\lambda)$ 就和 $p_{n-1}(\lambda)$ 异号，连号数少 1 个．同样，当 λ 值过 $p_n(\lambda)$ 的第二小的零点时，此时一定经过 $p_{n-1}(\lambda)$ 的最小零点，连号又少 1 个，也就是说，$\boldsymbol{C}$ 在 $[\lambda_0,+\infty)$ 内的零点个数为 $\varphi(\lambda_0)$ 的连号数.

实对称三对角矩阵特征值的对分法计算过程如下.

由定理 2 可知，凡经过 $p_n(x)$ 的一个零点，连号数 $\varphi(\lambda)$ 就少 1 个，因此利用定理 2 和圆盘定理，由对分法可将 $\boldsymbol{C}$ 的特征值隔离．下面是求最大特征值的所在邻域 $[m^*,M^*]$ 的步骤和算式.

① 给出 $\boldsymbol{C}$ 的 Sturm 序列：

$$\begin{cases}p_0(\lambda)=0\\ p_i(\lambda)=(c_i-\lambda)p_{i-1}(\lambda)-b_{i-1}^2p_{i-2}(\lambda)\end{cases}$$

② 由圆盘定理（推论）确定 $\boldsymbol{C}$ 的特征值的下界 m 和上界 M：

$$\begin{cases}m=\min(c_j-|b_j|-|b_{j-1}|)\\ M=\max(c_j+|b_j|+|b_{j-1}|)\end{cases}\qquad j=1,2,\cdots,n$$

③ 用对分法确定 m^* 和 M^*：

$$\begin{cases}\varphi(m^*)=1\\ \varphi(M)=0\end{cases}$$

计算过程如下：

$$M^*=M$$

$$\begin{cases}c_0=\dfrac{M+m}{2}\\ c_i=\dfrac{M+c_{i-1}}{2}\qquad \varphi(c_i)>1\quad i=1,2,\cdots\\ m^*=c_i\qquad\qquad\qquad \varphi(c_i)=1\end{cases}$$

上面用了对分法，下面继续用对分法确定范围更小的 m^* 和 M^*.

重新令 $c_0=\dfrac{m_0^*+M_0^*}{2}=\dfrac{m^*+M}{2}$，则

$$\begin{cases}\begin{cases}M^{(i)}=\begin{cases}M^{(i-1)} & \varphi(c^{(i-1)})=1\\ c^{(i-1)} & \varphi(c^{(i-1)})=0\end{cases}\\ m^{(i)}=\begin{cases}c^{(i-1)} & \varphi(c^{(i-1)})=1\\ m^{(i-1)} & \varphi(c^{(i-1)})=0\end{cases}\end{cases} & i=1,2,\cdots\\ c^{(i)}=\dfrac{M^{(i)}+m^{(i)}}{2} & M^{(i)}-m^{(i)}>\varepsilon\\ c=c^{(i)}\in[m^*,M^*]=[m^{(i)},M^{(i)}].\end{cases}$$

其中 c 即为所求.

若求第二大特征值的邻域，只须将式（5.3-3）中的 $\varphi(c^{(i-1)})=1$ 换成 $\varphi(c^{(i-1)})=2$，把 $\varphi(c^{(i-1)})=0$ 换成 $\varphi(c^{(i-1)})=1$．依此类推，可计算出任意大特征值的邻域.

2．计算实例

【例 1】 用对分法确定矩阵 $\boldsymbol{C}=\begin{bmatrix}1&2&0&0\\2&1&2&0\\0&2&1&2\\0&0&2&1\end{bmatrix}$ 的最大特征值，要求 $M^*-m^*\leqslant 5\times10^{-2}$.

【解】（1）本例的 Sturm 序列为

$$p_0(\lambda)=1$$
$$p_1(\lambda)=1-\lambda$$
$$p_2(\lambda)=(1-\lambda)p_1(\lambda)-4$$
$$p_3(\lambda)=(1-\lambda)p_2(\lambda)-4p_1(\lambda)$$
$$p_4(\lambda)=(1-\lambda)p_3(\lambda)-4p_2(\lambda)$$

（2）由圆盘定理推论可知 $\boldsymbol{C}$ 的特征值 λ 满足

$$1-2-2=-3\leqslant\lambda\leqslant1+2+2=5.$$

（3）通过连号数，用对分法确定 m^* 和 M^*.

计算结果如表 5.4 所示.

表 5.4 对分法的计算结果

λ	$p_0(\lambda)$	$p_1(\lambda)$	$p_2(\lambda)$	$p_3(\lambda)$	$p_4(\lambda)$	连 号 数
5	1	−4	12	−32	80	0
−3	1	4	12	32	80	4
1	1	1	0	−4	0	2
−1	1	2	0	−8	−16	3
3	1	1	−2	0	8	1

由计算结果可以看出 $\lambda_{\max} \in [3,5]$，即 $m^* = 3$，$M^* = 5$.

（4）用对分法计算 c.

计算结果如表 5.5 所示.

表 5.5　对分法的计算结果

λ	$p_0(\lambda)$	$p_1(\lambda)$	$p_2(\lambda)$	$p_3(\lambda)$	$p_4(\lambda)$	连　号　数
4	1	−3	5	−3	−11	1
4.5	1	−3.5	8.25	−14.875	19.062	0
4.25	1	−3.25	6.5625	−8.328125	0.8164	0
…	…	…	…	…	…	…
4.22	1	−3.22	6.3684	−7.626248	−0.9171	1

因为 $\varphi(4) = 1$，$\varphi(4.5) = 0$，所以最大特征值在 $[4,4.5]$ 内；因为 $\varphi(4.25) = 0$，所以最大特征值在 $[4,4.25]$ 内，最后的计算结果为 $\lambda_{\max} = 4.22$.

本例中间计算过程略.

对分法只能求出三对角实对称矩阵的特征值的近似值，无法求出特征向量. 为了求出更精确的特征值和特征向量，需要用到原点平移和逆幂法，这正是原点平移和逆幂法的真正用途.

求实对称矩阵 $\boldsymbol{A}$ 的第 i 大特征值的计算过程如下.

（1）用镜面反射变换将矩阵 $\boldsymbol{A}$ 转换成三对角矩阵 $\boldsymbol{A}^*$.

（2）计算 Sturm 序列：

取出对角线元素 c_i 及 $b2_i = b_i^2$（$i = 1,2,\cdots,n$）.

（3）计算

$$M = \max(c_i + |b_i| + |b_{i-1}|)，\quad i = 1,2,\cdots,n，\quad b_0 = 0.$$

$$m = \min(c_i - |b_i| - |b_{i-1}|)，\quad i = 1,2,\cdots,n，\quad b_0 = 0.$$

确定特征值的范围.

（4）用对分法找

$$\begin{cases} \varphi(M^*) = i - 1 \\ \varphi(m^*) = i \end{cases}$$

确定第 i 大特征值的所在邻域.

（5）用对分法确定符合精度的 λ_i 的近似值 $\tilde{\lambda}_i$.

（6）对原始矩阵做原点平移 $\boldsymbol{B} = \boldsymbol{A} - \tilde{\lambda}_i \boldsymbol{I}$.

（7）用逆幂法确定 $(\boldsymbol{B})^{-1}$ 的绝对值的第 i 大特征值 α_i.

（8）计算 $\boldsymbol{A}$ 的第 i 大特征值 λ_i：

$$\lambda_i = \tilde{\lambda}_i + 1/\alpha_i$$

习题 5

1．取 $\boldsymbol{x}^{(0)}=(1,1,1)^{\mathrm{T}}$，用幂法求方阵 $\boldsymbol{A}$ 的按模最大特征根及相应的特征向量：

$$\boldsymbol{A}=\begin{bmatrix}2 & 4 & 6\\ 3 & 9 & 15\\ 4 & 16 & 36\end{bmatrix}$$

2．已知对称三角方阵

$$\boldsymbol{A}=\begin{bmatrix}-2 & 1 & 0 & 0\\ 1 & -2 & 1 & 0\\ 0 & 1 & -2 & 1\\ 0 & 0 & 1 & -2\end{bmatrix}$$

问其在区间 $[-2,\ 0]$ 内有多少个特征根？

3．用对分法求对称三角方阵 $\boldsymbol{A}$ 的全部特征值，要求保留两位小数．

$$\boldsymbol{A}=\begin{bmatrix}1 & 2 & 0\\ 2 & -1 & 1\\ 0 & 1 & 3\end{bmatrix}$$

4．用 Householder 方法将矩阵

$$\boldsymbol{A}=\begin{bmatrix}1 & 3 & 4\\ 3 & 1 & 2\\ 4 & 2 & 1\end{bmatrix}$$

化为对称三对角方阵．

第 6 章　代数插值多项式

不少工程物理问题和科研问题都要涉及定积分，而能求出原函数的被积函数不多，能用 Newton-Leibnitz 公式算出积分值的函数也不多．导数、偏导数的理论计算虽然容易，但对于复杂函数及复合重数多的函数，实际计算并不容易．更加难以解决的是，无法编写出通用程序；若函数本身是用仪器仪表所测得的离散点的值，则处理更不方便．为了解决这一问题，只能用简单的性质良好的函数代替复杂函数或图表表述的函数，其中多项式的性质最为良好，因此常以多项式代替被逼近函数．当然，三角函数、有理函数的性质也不错，因此也常用这些函数来代替复杂函数，这就是函数逼近．

函数逼近有两条途径，一是插值逼近，二是非插值逼近．本书只介绍插值逼近．

所有的插值逼近都有一个共同的特点，也是其基本特征，那就是选取一些特殊点，这些点的函数值甚至导数值、高阶导数值已知，用以插值的函数和被逼近的函数在这些点的函数值（和导数值等）都相等，这些点称为节点，当插值函数为多项式时，称为代数插值多项式．本章介绍 Lagrange 插值多项式、Newton 插值多项式和幂级数型插值多项式．

6.1　Lagrange 插值多项式

n 阶 Lagrange 插值多项式由 $n+1$ 个对称的 n 次多项式组成，Lagrange 插值多项式用于数值积分，是数值积分的基础．

6.1.1　Lagrange 插值多项式

1．一阶 Lagrange 插值多项式

对于连续函数 $y=f(x)$，$x\in[a,b]$，取节点为 a 和 b，若用一阶 Lagrange 插值多项式 $L_1(x)$ 逼近 $f(x)$（常以 $x_0=a$ 和 $x_1=b$），则

$$L_1(x)=A(x-x_1)+B(x-x_0)\,.$$

由插值多项式的性质有

$$L_1(x_0)=f(a)=y_0\Rightarrow A=\frac{f(x_0)}{x_0-x_1}=\frac{y_0}{x_0-x_1}$$

$$L_1(x_1)=f(b)=y_1\Rightarrow B=\frac{f(x_1)}{x_1-x_0}=\frac{y_1}{x_1-x_0}$$

由此有

$$L_1(x)=\frac{x-x_1}{x_0-x_1}y_0+\frac{x-x_0}{x_1-x_0}y_1$$

2．二阶 Lagrange 插值多项式

常取节点 $x_0=a$，$x_1\in(a,b)$，$x_2=b$，令二阶 Lagrange 插值多项式为

$$L_2(x)=A(x-x_1)(x-x_2)+B(x-x_0)(x-x_2)+C(x-x_0)(x-x_1)$$

由插值多项式的性质有

$$L_2(x_0)=y_0\Rightarrow A=\frac{y_0}{(x_0-x_1)(x_0-x_2)}$$

$$L_2(x_1)=y_1\Rightarrow B=\frac{y_1}{(x_1-x_0)(x_1-x_2)}$$

$$L_2(x_2)=y_2\Rightarrow C=\frac{y_2}{(x_2-x_0)(x_2-x_1)}$$

$$L_2(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x-x_2)}y_0+\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1+\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2$$

3．n 阶 Lagrange 插值多项式

由一阶、二阶 Lagrange 插值多项式可以看出，n 阶 Lagrange 插值多项式由 $n+1$ 个形态对称的 n 次多项式（的和）组成，其形式可表述成

$$\begin{cases}L_n(x)=\sum\limits_{i=0}^{n}A_i(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)\\ A_i=\dfrac{y_i}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}\end{cases}\tag{6.1-1}$$

式（6.1-1）还可以表述为

$$\begin{cases}L_n(x)=\sum\limits_{i=0}^{n}l_i(x)y_i\\ l_i(x)=\dfrac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}\\ \qquad=\prod\limits_{\substack{j=0\\ j\neq i}}^{n}\dfrac{x-x_j}{x_i-x_j}\end{cases}\tag{6.1-2}$$

由式（6.1-1）和式（6.1-2）都可得到

$$L_n(x_i)=l_i(x_i)y_i=y_i,\quad i=0,1,\cdots,n$$

所以式（6.1-1）或式（6.1-2）是插值多项式.

6.1.2　代数插值多项式余项

用 n 阶代数插值多项式逼近 $f(x)$ 后，逼近效果如何？设有 n 阶代数插值多项式 $p_n(x)$ 和其要逼近的函数 $f(x)$，$f(x)-p_n(x)$ 为多少？

【定理】设函数 $f(x)$ 在 $x\in[a,b]$ 上至少有 $n+1$ 阶连续导数，$p_n(x)$ 是 n 阶代数插值多项式，节点 $x_i\in[a,b]$（$i=0,1,\cdots,n$），则

$$f(x)-p_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

我们称 $R_n(x)=f(x)-p_n(x)$ 为插值余项，上式中 ξ 在 $x_0,x_1,\cdots,x_n$ 之间.

【证明】作辅助函数

$$\varphi(t)=f(t)-p_n(t)-\frac{f(x)-p_n(x)}{(x-x_0)(x-x_1)\cdots(x-x_n)}(t-x_0)(t-x_1)\cdots(t-x_n)$$

显然

$$\begin{cases}\varphi(x_i)=0, & i=0,1,\cdots,n\\ \varphi(x)=0\end{cases}$$

即 $\varphi(t)$ 有 $n+2$ 个零点，根据罗尔定理，存在 $\xi_1,\xi_2,\cdots,\xi_{n+1}$ 使得 $\varphi'(\xi_i)=0$ $(i=1,2,\cdots,n+1)$. 反复利用罗尔定理可知，在 $x_0,x_1,\cdots,x_n$ 之间存在 ξ 使得 $\varphi^{(n+1)}(\xi)=0$，即

$$f^{(n+1)}(\xi)-p_n^{(n+1)}(\xi)-\frac{f(x)-p_n(x)}{(x-x_0)\cdots(x-x_n)}((t-x_0)\cdots(t-x_n))^{(n+1)}$$

$$=f^{(n+1)}(\xi)-\frac{f(x)-p_n(x)}{(x-x_0)\cdots(x-x_n)}(n+1)!=0$$

由此得

$$f(x)-p_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

【推论】若 $f(x)$ 是 n 阶多项式，则

$$p_n(x)=f(x).$$

上面的定理不是只针对 Lagrange 插值多项式的，在证明该定理时，只用了代数插值多项式的性质，因而上面的定理对任意插值多项式均成立.

6.1.3　Lagrange 插值多项式计算及计算实例

由式（6.1-1）可以看出，要计算 Lagrange 插值多项式，最好先计算系数 A_i，然后任给 x 值，计算 $L_n(x)$ 的值.

【例 1】求一个不高于 3 次的 Lagrange 插值多项式 $L_3(x)$，使 $L_3(0)=0$，$L_3(1)=1$，$L_3(2)=3$，$L_3(3)=3$，并求 $L_3(1.5)$ 的值.

【解】本例所选节点 $x_0=0$，$x_1=1$，$x_2=2$，$x_3=3$，表 6.1 记录了节点值和函数值.

表 6.1 算例节点及函数值

x	0	1	2	3
y	0	1	3	3

按式（6.1-1）计算 A_0, A_1, A_2, A_3，结果如下：

$$A_0=\frac{f(x_0)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}=\frac{0}{(0-1)(0-2)(0-3)}=0$$

$$A_1=\frac{f(x_1)}{(x_1-x_0)(x_1-x_2)(x_0-x_3)}=\frac{1}{(1-0)(1-2)(1-3)}=0.5$$

$$A_2=\frac{f(x_2)}{(x_2-x_1)(x_2-x_2)(x_2-x_3)}=\frac{3}{(2-0)(2-1)(2-3)}=-1.5$$

$$A_3=\frac{f(x_3)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}=\frac{3}{(3-0)(3-1)(3-2)}=0.5$$

$$\begin{aligned}L_3(1.5)&=A_0(x-x_1)(x-x_2)(x-x_3)+A_1(x-x_0)(x-x_2)(x-x_3)+\\&\quad A_2(x-x_0)(x-x_1)(x-x_3)+A_3(x-x_0)(x-x_1)(x-x_2)\\&=0.5(1.5-0)(1.5-2)(1.5-3)-1.5(1.5-0)(1.5-1)(1.5-3)+0.5(1.5-0)(1.5-1)(1.5-2)\\&=2.0625\end{aligned}$$

6.2 Newton 插值多项式

Newton 插值多项式是另一种形式的插值多项式．n 次 Newton 插值多项式由 $n+1$ 个次数不同的多项式组成，其一般格式为

$$N_n(x)=c_0+c_1(x-x_0)+\cdots+c_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})=\sum_{i=0}^{n}c_i\prod_{j=0}^{i-1}(x-x_i)$$

式中 x_i 为节点．

6.2.1 一阶、二阶 Newton 插值多项式系数计算

1．一阶 Newton 插值多项式

设有函数 $y=f(x)$，选 x_0，x_1 为节点，则一阶 Newton 插值多项式为

$$N_1(x)=c_0+c_1(x-x_0)$$

由插值函数的定义有

$$N_1(x_0)=y_0\Rightarrow c_0=y_0,\quad N_1(x_1)=y_1\Rightarrow c_1=\frac{y_1-c_0}{x_1-x_0}=\frac{y_1-y_0}{x_1-x_0},$$

即

$$N_1(x)=y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0)\,.$$

2．二阶 Newton 插值多项式

设有函数 $y=f(x)$，选 x_0,x_1,x_2 为节点，则二阶 Newton 插值多项式为

$$N_2(x)=c_0+c_1(x-x_0)+c_2(x-x_0)(x-x_1)$$

由插值函数的定义有

$$N_2(x_0)=y_0\Rightarrow c_0=y_0,\quad N_2(x_1)=y_1\Rightarrow c_1=\frac{y_1-c_0}{x_1-x_0}=\frac{y_1-y_0}{x_1-x_0}$$

$$N_2(x_2)=y_2\Rightarrow c_2=\frac{y_2-y_0-\dfrac{y_1-y_0}{x_1-x_0}(x_2-x_0)}{(x_2-x_0)(x_2-x_1)}=\frac{\dfrac{y_2-y_1}{x_2-x_1}-\dfrac{y_1-y_0}{x_1-x_0}}{x_2-x_0}$$

即

$$N_2(x)=y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0)+\frac{\dfrac{y_2-y_1}{x_2-x_1}-\dfrac{y_1-y_0}{x_1-x_0}}{x_2-x_0}(x-x_0)(x-x_1)\,.$$

由二阶 Newton 插值多项式和一阶 Newton 插值多项式的系数可以看出，后者的零次系数和一次系数与前者一样，这一性质将一直保持下去，这是称 Newton 插值多项式为逐步插值多项式的原因.

Lagrange 插值多项式各系数之间是独立的，计算公式简单．而 Newton 插值多项式系数之间是有关联的，为了计算 n 阶 Newton 插值多项式的各系数，需要做以下准备工作.

6.2.2　差商及其计算公式

【定义】 令 $f[x_i]=f(x_i)$ 为 $f(x)$ 在 x_i 处的零阶差商，而

$$f[x_i,x_{i+1},\cdots,x_{i+p}]=\frac{f[x_{i+1},x_{i+1},\cdots,x_{i+p}]-f[x_i,x_{i+1},\cdots,x_{i+p-1}]}{x_{i+p}-x_i}$$

为 $f(x)$ 在 x_i 处的 p 阶差商.

【定理 1】 $f(x)$ 在 $x=x_0$ 处的 p 阶差商是 $f(x_0),f(x_1),\cdots,f(x_p)$ 的线性组合，且满足

$$f[x_0,x_1,\cdots,x_p]=\sum_{i=0}^{p}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_p)}\,.$$

【证明】 现用归纳法证明这一定理.

当 $p=1$ 时，

$$f[x_0,x_1]=\frac{f[x_1]-f[x_0]}{x_1-x_0}=\frac{f(x_1)-f(x_0)}{x_1-x_0}=\frac{f(x_0)}{x_0-x_1}+\frac{f(x_1)}{x_1-x_0}\,,$$

定理成立.

设 $p=m-1$ 时定理成立，即

$$f[x_0,x_1,\cdots,x_{m-1}]=\sum_{i=0}^{m-1}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{m-1})},$$

$$f[x_1,x_2,\cdots,x_m]=\sum_{i=1}^{m}\frac{f(x_i)}{(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_m)}.$$

当 $p=m$ 时，由 p 阶差商的定义有

$$f[x_0,x_1,\cdots,x_m]=\frac{f[x_1,x_2,\cdots,x_m]-f[x_0,x_1,\cdots,x_{m-1}]}{x_m-x_0}$$

$$\begin{aligned}
f[x_1,x_2,\cdots,x_m]&=\sum_{i=1}^{m}\frac{f(x_i)}{(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_m)}\\
&=\frac{f(x_m)}{(x_m-x_1)\cdots(x_m-x_{m-1})}+\sum_{i=1}^{m-1}\frac{f(x_i)(x_i-x_0)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_m)}\\
f[x_0,x_1,\cdots,x_{m-1}]&=\sum_{i=0}^{m-1}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{m-1})}\\
&=\frac{f(x_0)}{(x_0-x_1)\cdots(x_0-x_{m-1})}+\sum_{i=1}^{m-1}\frac{f(x_i)(x_i-x_m)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_m)}\\
f[x_0,x_1,\cdots,x_m]&=\frac{f(x_m)}{(x_m-x_0)\cdots(x_m-x_{m-1})}-\frac{f(x_0)}{(x_0-x_1)\cdots(x_0-x_{m-1})(x_m-x_0)}+\\
&\quad\sum_{i=1}^{m-1}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_m)}\cdot\frac{(x_i-x_0)-(x_i-x_m)}{x_m-x_0}\\
&=\frac{f(x_m)}{(x_m-x_0)\cdots(x_m-x_{m-1})}+\frac{f(x_0)}{(x_0-x_1)\cdots(x_0-x_m)}+\\
&\quad\sum_{i=1}^{m-1}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_m)}\\
&=\sum_{i=0}^{m}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_m)}
\end{aligned}$$

满足定理结论，也就是说定理成立.

【定理 2】 设 $y_i=f(x_i)$，$i=0,1,\cdots,n$，x_i 互异.

$$\begin{cases}N_n(x)=c_0+c_1(x-x_0)+\cdots+c_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})\\ c_i=f[x_0,x_1,\cdots,x_i], \qquad i=0,1,\cdots n\end{cases}\tag{6.2-1}$$

则 $N_n(x_i)=f(x_i)=y_i$，即 $N_n(x)$ 是 n 阶代数插值多项式.

【证明】 设 $L_n(x)$ 是以 x_0，x_1，$\cdots$，x_n 为节点的 n 阶 Lagrange 插值多项式，且令 $L_0(x)=f(x_0)=y_0$，$L_k(x)$ 是以 $x_0,x_1,\cdots,x_k$ 为节点的 k 阶 Lagrange 插值多项式，则

$$L_n(x)=L_0(x)+[L_1(x)-L_0(x)]+\cdots+[L_n(x)-L_{n-1}(x)]$$

显然，$L_k(x)-L_{k-1}(x)$ 是 k 次多项式，且满足

$$L_k(x_i)-L_{k-1}(x_i)=f(x_i)-f(x_i)=0,\quad i=0,1,\cdots,k-1.$$

可令

$$L_k(x_i)-L_{k-1}(x_i)=A_k(x-x_0)(x-x_1)\cdots(x-x_{k-1})$$

当 $x=x_k$ 时，$L_k(x_k)=f(x_k)$.

$$\begin{aligned}A_k&=\frac{f(x_k)-L_{k-1}(x_k)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})}\\&=\frac{f(x_k)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})}-\frac{\sum_{i=0}^{k-1}f(x_i)\dfrac{(x_k-x_0)\cdots(x_k-x_{i-1})(x_k-x_{i+1})\cdots(x_k-x_{k-1})}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{k-1})}}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})}\\&=\sum_{i=0}^{k}\frac{f(x_k)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_k)}=f[x_0,x_1,\cdots,x_k]\end{aligned}$$

所以

$$\begin{aligned}N_n(x)=L_n(x)&=f[x_0]+f[x_0,x_1](x-x_0)+\cdots+f[x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})\\&=c_0+c_1(x-x_0)+\cdots+c_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})\\&=\sum_{i=0}^{n}c_i\prod_{j=0}^{i-1}(x-x_j)\end{aligned}$$

由此可以看出，n 阶 Newton 插值多项式系数 c_i 的数学意义是函数 $f(x)$ 在 x_0 点的 i 阶差商.

6.2.3　Newton 插值多项式计算

和 Lagrange 插值多项式做插值计算一样，用 Newton 插值多项式做插值计算分两步进行.

1．Newton 插值多项式系数计算

Newton 插值多项式有 $n+1$ 个系数，这些系数分别是 $f(x)$ 在 x_0 点的 0～n 阶差商．要计算 $f(x)$ 在 x_0 点的 n 阶差商 $f[x_0,x_1,\cdots,x_n]$，需要计算两个 $n-1$ 阶差商 $f[x_0,x_1,\cdots,x_{n-1}]$ 和 $f[x_1,x_2,\cdots,x_n]$，还需要计算 3 个 $n-2$ 阶差商，最后需要计算 n 个一阶差商 $f[x_0,x_1]$，$f[x_1,x_2],\cdots$，$f[x_{n-1},x_n]$，这些差商形成一个差商表．本书将差商表列成二维表（为了和其他书中的表有所区别，称本书中的表为改进差商表）并给出了另一简洁递推公式．当 $n=3$ 时，差商表见表 6.2.

表 6.2　$n=3$ 时的差商表

i	x_i	$f[x_i]$	$f[x_i,x_{i+1}]$	$f[x_i,x_{i+1},x_{i+2}]$	$f[x_i,x_{i+1},x_{i+2},x_{i+3}]$
0	x_0	$f[x_0]$	$f[x_0,x_1]$	$f[x_0,x_1,x_2]$	$f[x_0,x_1,x_2,x_3]$
1	x_1	$f[x_1]$	$f[x_1,x_2]$	$f[x_1,x_2,x_3]$	
2	x_2	$f[x_2]$	$f[x_2,x_3]$		
3	x_3	$f[x_3]$			

令

$$\begin{cases} c_{i0} = f[x_i] & i = 0,1,\cdots,n \\ c_{i,j} = f[x_i, x_{i+1}, \cdots, x_{i+j}] & i = 0,1,\cdots,n-j;\ j = 1,2,\cdots,n \end{cases},$$

由差商的定义有

$$\begin{aligned} c_{i,j} &= f[x_i, x_{i+1}, \cdots, x_{i+j}] \\ &= \frac{f[x_{i+1}, x_{i+2}, \cdots, x_{i+1+j-1}] - f[x_i, x_{i+1}, \cdots, x_{i+j-1}]}{x_{i+j} - x_i}, \quad \begin{matrix} j = 1,2,\cdots,n \\ i = 0,1,\cdots,n-j \end{matrix} \\ &= \frac{c_{i+1,j-1} - c_{i,j-1}}{x_{i+j} - x_i} \end{aligned} \tag{6.2-2}$$

式（6.2-2）为差商递推计算公式，即 Newton 插值多项式系数递推公式.

2．Newton 插值多项式的值的计算

手算时可按式（6.2-1）直接计算 Newton 插值多项式的值，但由于插值是逼近，是对整个函数逼近，也就是说要计算相当多的点的 $N_n(x)$ 值，因此须考虑计算量.

对于 $N_n(x) = c_0 + c_1(x - x_0) + \cdots + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1})$，仿秦九韶法，可将式（6.2-1）改写成

$$\begin{aligned} &c_0 + c_1(x - x_0) + \cdots + c_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) \\ &= (\cdots(c_n(x - x_{n-1}) + c_{n-1}) \bullet (x - x_{n-2}) + \cdots + c_1) \bullet (x - x_0) + c_0 \end{aligned} \tag{6.2-3}$$

式（6.2-3）和程序设计语言无对应关系，现将之改写成递推公式：

$$\begin{cases} S_n = c_n \\ S_i = S_{i+1}(x - x_i) + c_i \end{cases}, \quad i = n-1, n-2, \cdots, 0 \tag{6.2-4}$$

式（6.2-4）和 C 语言的以下语句对应：

```
S[n]=c[0][n]
for(i=n-1;i>=0;i--)
    S[i]=S[i+1]*(z-x[i])+c[i]
```

程序段中的 z 是式（6.2-4）中的 x.

直接按式（6.2-1）计算 Newton 插值多项式值需要用 $\frac{1}{2}n^2$ 次乘法，按式（6.2-4）计算只用 n 次乘法．由于式（6.2-4）是借助秦九韶算法来计算多项式值的算法，故称为仿秦九韶法.

6.2.4 用 Newton 插值多项式做插值计算的计算步骤和实例

计算步骤如下.

（1）给出节点的自变量值和节点函数值；

（2）利用二元递推公式：

$$c_{i,j}=\frac{c_{i+1,j-1}-c_{i,j-1}}{x_{i+j}-x_i}\qquad \begin{matrix} j=1,2,\cdots,n \\ i=0,1,\cdots,n-j \end{matrix}$$

生成改进差商表；

（3）从差商表中取出 c_{ij} 生成 Newton 插值多项式；

（4）对于工程物理问题，用仿秦九韶法做插值计算（若是手算少数点的值，可不用仿秦九韶法）.

【例 1】求一个阶数不高于 4 阶的 Newton 插值多项式，使 $N_4(0)=1$，$N_4(1)=1$，$N_4(2)=2$，$N_4(3)=3$，$N_4(4)=3$，并计算 $N_4(3.5)$ 的值.

【解】设插值多项式为

$$N_4(x)=\sum_{i=0}^{4}c_{i0}\prod_{j=0}^{i-1}(x-x_j)$$

$$c_{ij}=\frac{c_{i+1,j-1}-c_{i,j}}{x_{i+j}-x_i},\quad j=1,2,\cdots,n,\quad i=0,1,\cdots,n-j$$

计算结果见表 6.3.

表 6.3　例 1 的计算结果

i	x_i	y_i	c_{i1}	c_{i2}	c_{i3}	c_{i4}
0	0	1	0	0.5	$-1/6$	0
1	1	1	1	0	$-1/6$	
2	2	2	1	0.5		
3	3	3	0			
4	4	3				

表 6.3 称为改进差商表，为便于比较，特附上传统差商表，如表 6.4 所示.

表 6.4　传统差商表

i	x_i	y_i				
0	0	1				
			0			
1	1	1		0.5		
			1		$-\frac{1}{6}$	
2	2	2		0		0
			1		$-\frac{1}{6}$	
3	3	3		−0.5		
			0			
4	4	3				

由此可得

$$N_4(x)=1+0.5x(x-1)-\frac{1}{6}x(x-1)(x-2)\ .$$

$$N_4(3.5)=1+0.5\times3.5\times(3.5-1)-\frac{1}{6}\times3.5\times(3.5-1)\times(3.5-2)=1+3.8750-0.96875\doteq3.90625$$

6.2.5 带重节点的 Newton 插值多项式

【定义】若节点的函数值已知，节点的导数值也已知，则称该节点为一重节点，简称重节点．同样可定义 k 重节点为函数值及 $1\sim k-1$ 阶导数值已知的节点．

作插值多项式时，若遇有重节点，则插值多项式在该节点的值及导数值应等于该点的函数值和导数值．

本书不考虑多重节点．

带重节点的 Newton 插值多项式和一般 Newton 插值多项式一样，其系数的计算公式为

$$c_{ij}=\begin{cases}f'(x_i), & j=1,f'(x_i)\text{已知}\\ \dfrac{c_{i+1,j-1}-c_{i,j-1}}{x_{i+j}-x_i}, & f'(x_i)\text{未知}\end{cases},\quad\begin{matrix}j=1,2,\cdots,n\\ i=0,1,\cdots n-j\end{matrix}\qquad(6.2\text{-}5)$$

由式（6.2-5）可以看出，带重节点的 Newton 插值多项式系数计算公式基本和不带重节点的一样，只是一阶差商部分当 $f'(x_i)$ 已知时，用 $f'(x_i)$ 代替一阶差商，而不再计算．

计算带重节点的 Newton 插值多项式时，一个重节点占据两个标号，这两个标号的自变量（节点）值和函数值都是同一点的值，当然都相等．

6.2.6 带重节点的 Newton 插值多项式计算过程和计算实例

带重节点的 Newton 插值多项式和不带重节点的 Newton 插值多项式的计算过程相同．

【例 2】求一个次数不高于 4 次的 Newton 插值多项式 $N_4(x)$，使 $N_4(0)=1$，$N_4(1)=2$，$N_4'(1)=2$，$N_4(2)=2$，$N_4'(2)=3$，并计算 $N_4(1.5)$ 的值．

【解】设插值多项式为

$$N_4(x)=\sum_{i=0}^{4}c_i\prod_{j=0}^{i-1}(x-x_j)$$

（1）利用式（6.2-5）即

$$c_{ij}=\begin{cases}f'(x_i), & j=1,f'(x_i)\text{ 已知}\\ \dfrac{c_{i+1,j-1}-c_{i,j-1}}{x_{i+j}-x_i}, & f'(x_i)\text{ 未知, }i=0,1,\cdots,n-j\end{cases}$$

得出差商表，如表 6.5 所示．

表 6.5　差商表

i	x_i	y_i	c_{i1}	c_{i2}	c_{i3}	c_{i4}
0	0	1	1	1	−1.5	3.25
1	1	2	2	−2	5	
2	1	2	0	3		
3	2	2	3			
4	2	2				

带下画线的数字是已知的（导数），不用计算，直接利用即可.

（2）从差商表第 0 行取出 c_{0j}，给出 Newton 插值多项式：

$$N_4(x)=1+x+x(x-1)-1.5x(x-1)^2+3.25x(x-1)^2(x-2)$$

（3）（手算）代入所给点值，计算 Newton 插值多项式：

$$\begin{aligned}N_4(1.5)&=1+1.5+1.5(1.5-1)-1.5\times1.5\times(1.5-1)^2+3.25\times1.5\times(1.5-1)^2(1.5-2)\\&=2.087125\end{aligned}$$

注意：手算 Newton 插值多项式的值时，不应将算式改写成幂级数，也不必将之转换成式（6.2-4），因为做了转换后不便于检验计算是否正确.

6.2.7　带重节点的插值多项式的插值余项

当插值多项式 $P_n(x)$ 带一个重节点时，$f(x)-P_n(x)=?$ 这也是人们关心的问题.

【定理】 若函数 $y=f(x)$ 在区间 $[a,b]$ 上至少有 $n+1$ 阶连续导数，$P_n(x)$ 是以 x_i（$i=0,1,\cdots,n$）为节点的插值多项式，其中 $x_{n-1}=x_n$ 为重节点，则

$$f(x)-P_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)\cdots(x-x_{n-2})(x-x_{n-1})^2\ .$$

【证明】 作辅助函数

$$\varphi(t)=f(t)-P_n(t)-\frac{(f(x)-P_n(x))(t-x_0)\cdots(t-x_{n-2})(t-x_{n-1})^2}{(x-x_0)\cdots(x-x_{n-2})(x-x_{n-1})^2}$$

显然，

$$\begin{cases}\varphi(x_i)=0, & i=0,1,\cdots,n-1\\ \varphi(x)=0\end{cases}$$

即 $\varphi(t)$ 至少有 $n+1$ 个零点，其中 $\varphi(x_{n-1})$ 是二重零点，由 Roll 定理可知，$\varphi'(t)$ 有 $n+1$ 个零点.

反复利用 Roll 中值定理有，存在 ξ，使得 $\varphi^{(n+1)}(\xi)=0$，即

$$f^{(n+1)}(\xi)-P_n^{(n+1)}(\xi)+\frac{(f(x)-P_n(x))(n+1)!}{(x-x_0)(x-x_1)\cdots(x-x_{n-1})^2}=0$$

$$f(x)-P_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_{n-1})^2$$

当插值多项式有更多的重节点时，导数函数仍有 $n+1$ 个零点，仍有上面类似的结论.

6.3 幂级数型代数插值多项式

多项式是所有函数中性质最良好的函数，幂级数又是多项式中计算最方便、计算量最小的子类，用幂级数型插值多项式逼近函数具有鲜见的优良特性.

6.3.1 幂级数型插值多项式

作插值多项式时节点的选取及节点的编号是人为的. 节点选取原则和节点编号的目的是，使得生成插值多项式时方便，计算公式简洁，且插值多项式的性质良好.

为了节省篇幅，这里只介绍两种幂级数型插值多项式.

1. 等距节点幂级数型插值多项式

对于 $y=f(x)$， $x\in[a,b]$，选取 $x_0=a$， $x_i=a+ih$， $i=0,1,\cdots,n$， $h=\dfrac{b-a}{n}$.

令 $z_n(x)$ 为 n 阶幂级数插值多项式，则

$$\begin{cases} z_n(x)=\sum\limits_{i=0}^{n}\alpha_i(x-x_0)^i=\sum\limits_{i=0}^{n}\alpha_i\overline{x}^i=z_n(\overline{x}) \\ z_n(\overline{x}_j)=\sum\limits_{i=0}^{n}\alpha_i\overline{x}_j^{\,i}=\sum\limits_{i=0}^{n}\alpha_j(jh)^i=f(x_j)=y_j,\ j=0,1,\cdots,n \end{cases} \tag{6.3-1}$$

可以通过以下过程计算 α_i（$i=0,1,\cdots,n$），显然

$$\begin{cases} \alpha_0=y_0 \\ \sum\limits_{i=1}^{n}\alpha_i(x_j-a_0)^{i-1}=\sum\limits_{i=1}^{n}\alpha_i(jh)^i=(y_j-\alpha_0)\big/jh=y_j^{(1)},\ j=1,2,\cdots,n \end{cases} \tag{6.3-2}$$

作 $n-1$ 阶 Lagrange 插值多项式 $L_{n-1}(x)$，使

$$\begin{aligned} L_{n-1}(x)&=\sum_{j=1}^{n}\frac{(x-x_1)(x-x_2)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_j-x_1)(x_j-x_2)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}y_j^{(1)} \\ &=\sum_{j=1}^{n}\frac{(\overline{x}-h)(\overline{x}-2h)\cdots(\overline{x}-(j-1)h)(\overline{x}-(j+1)h)\cdots(\overline{x}-nh)}{(j-1)(j-2)\cdots(-1)(-2)\cdots(-(n-j))h^{n-1}} \end{aligned}$$

由此可得

$$\alpha_1=\sum_{j=1}^{n}(-1)^{j+1}C_n^j y_j^{(1)} \tag{6.3-3}$$

同样可得

$$\begin{cases} \alpha_k = \sum_{j=1}^{n-k+1} (-1)^{j+1} C_{n-k+1}^{j} y_j^{(k)} \\ y_j^{(k)} = (y_j^{(k-1)} - \alpha_{k-1}) \big/ jh \\ y_j^{(0)} = y_j \end{cases} \quad \begin{matrix} k = 1,2,\cdots,n \\ j = 1,2,\cdots,n-k \end{matrix} \tag{6.3-4}$$

式（6.3-4）为等距节点幂级数型插值多项式系数的计算公式.

2．平方等距节点幂级数型插值多项式

对于函数 $y = f(x)$，$x \in [a,b]$，取节点 $x_0 = \dfrac{a+b}{2}$，$x_{2i-1} = x_0 + \sqrt{ih}$，$x_{2i} = x_0 - \sqrt{ih}$，$i = 1,2,\cdots,m$，$h = \dfrac{(b-a)^2}{4m}$．则幂级数型代数插值多项式

$$z_{2m}(x) = \sum_{i=0}^{2m} \alpha_i (x - x_0)^i$$

的系数计算公式为

$$\begin{cases} \alpha_0 = y_0 \\ \alpha_{2k-1} = \sum_{j=1}^{m-k+1} (-1)^{j+1} C_{m-k+1}^{j} u_j^{(k)} \\ u_j^{(k)} = \dfrac{u_j^{(k-1)} - \alpha_{2k-3}}{jh} \\ u_j^{(1)} = \dfrac{y_{2j-1} - y_{2j}}{2\sqrt{jh}} \\ \alpha_{2k} = \sum_{j=1}^{m-k+1} (-1)^{j+1} C_{m-k+1}^{j} v_j^{(k)} \\ v_j^{(k)} = \dfrac{v_j^{(k-1)} - \alpha_{2k-2}}{jh} \\ v_j^{(1)} = \dfrac{y_{2j-1} + y_{2j} - 2y_0}{2jh} \end{cases} \quad \begin{matrix} k = 1,2,\cdots,m \\ j = 1,2,\cdots,m-j+1 \end{matrix} \tag{6.3-5}$$

实际上当 $z(x)$ 是代数插值多项式时，必须满足

$$\sum_{i=0}^{2m} \alpha_i (x_j - x_0)^i = y_j, \quad j = 0,1,\cdots,2m \tag{6.3-6}$$

由此可以得到

$$\alpha_0 = y_0$$

将方程组（6.3-6）编号为 1 和编号为 2 的方程相加……将编号为奇数的方程和编号为偶数的（大一个序号）方程相加和相减，化简，并将 $\alpha_0 = y_0$ 代入则得

$$\begin{matrix}\sum_{k=1}^{m}\alpha_{2k-1}(jh)^k = u_j^{(1)} \\ \sum_{k=1}^{m}\alpha_{2k}(jh)^k = v_j^{(1)}\end{matrix} \quad , \quad j=1,2,\cdots,m$$

仿式（6.3-4）则可得到式（6.3-5）.

6.3.2 幂级数型插值多项式计算过程和计算实例

1．计算过程

对于等距插值，其计算过程如下.

（1）利用杨辉三角公式计算组合数：

$$\begin{cases}C_k^k = C_k^0 = 1, & k=1,2,\cdots,n \\ C_k^i = C_{k-1}^i + C_{k-1}^{i-1}, & i=1,2,\cdots,k-1\end{cases} \tag{6.3-7}$$

式中 n 为插值多项式的阶.

（2）利用式（6.3-4）计算幂级数型插值多项式的系数 α_i .

（3）利用秦九韶法计算插值多项式的值：

$$\begin{cases}S_n = \alpha_n \\ S_i = S_{i+1}(x-a) + \alpha_i & i=n-1,n-2,\cdots,0\end{cases}$$

对于平方等距插值多项式，其计算过程如下.

（1）利用杨辉三角公式计算组合数：

$$\begin{cases}C_k^k = C_k^0 = 1, & k=1,2,\cdots,m \\ C_k^i = C_{k-1}^i + C_{k-1}^{i-1}, & i=1,2,\cdots,k-1\end{cases} \tag{6.3-8}$$

式中 $m=\dfrac{n}{2}$，为插值多项式的阶.

（2）利用式（6.3-5）计算幂级数型插值多项式的系数 α_{2i-1} 和 α_{2i} .

（3）利用秦九韶法计算插值多项式的值：

$$\begin{cases}S_n = S_{2m} = \alpha_{2m} \\ S_i = S_{i+1}\left(x - \dfrac{a+b}{2}\right) + \alpha_i, & i=n-1,n-2,\cdots,0\end{cases}$$

2．计算实例

【例 1】 用一个不高于 2 次的幂级数插值多项式逼近 $\ln x$，节点值取 $x_0 = 1.2$，$x_1 = 1.4$，$x_2 = 1.6$，并计算 $z_2(1.5)$ 的值.

【解】 节点处的函数值为

x	1.2	1.4	1.6
y	0.1823216	0.3364726	0.4700036

按式（6.3-4）得

$$a_0 = y_0 = 0.1823216$$

$$y_1^{(1)} = \frac{y_1 - y_0}{h} = 0.7707530$$

$$y_2^{(1)} = \frac{y_2 - y_0}{2h} = 0.7192050$$

$$a_1 = C_2^1 y_1^{(1)} - C_2^2 y_2^{(1)} = 2 \times 0.7707530 - 0.7192050 = 0.8223010$$

$$y_1^{(2)} = \frac{y_1^{(1)} - a_1}{h} = -0.2577400$$

$$a_2 = C_1^1 y_1^{(2)} = -0.2577400$$

由此得

$$z_2(x) = 0.1823216 + 0.8223010(x - 1.2) - 0.257740(x - 1.2)^2$$

$$z_2(1.5) = 0.405815$$

【例 2】同例 1，利用平方等距插值多项式计算例 1 的值.

【解】按平方等距插值多项式要求

$$x_0 = 1.4，\ \sqrt{h} = 0.2，\ h = 0.04$$

节点号及对应函数值为

i	0	1	2
x	1.4	1.6	1.2
y	0.3364722	0.4700036	0.1823216

按式（6.3-5）计算的系数值为

$$\alpha_0 = 0.3364722$$

$$u_1^{(1)} = \frac{y_1 - y_2}{2\sqrt{h}} = 0.7192050$$

$$v_1^{(1)} = \frac{y_1 + y_2 - 2y_0}{2h} = -0.2577400$$

$$\alpha_1 = C_1^1 u_1^{(1)} = 0.7192050$$

$$\alpha_2 = C_1^1 v_1^{(1)} = -0.2577400$$

所以

$$z_2(x) = 0.3364722 + 0.7192050(x - 1.4) - 0.257740(x - 1.4)^2$$

$$z_2(1.5) = 0.405815$$

因平方等距插值多项式的计算量小，以后算例皆只用平方等距插值多项式.

6.4 代数插值多项式的收敛性和稳定性

6.4.1 代数插值多项式的收敛性

以同一组节点所生成的同阶代数插值多项式都是相等的，因此各种代数插值多项式的收敛性都是相同的.

【引理】 $\sum_{i=1}^{n}(-1)^{i+1}C_n^i i^k=\begin{cases}1, & k=0\\ 0, & k=1,2,\cdots,n-1\end{cases}$.

【证明】 当 $k=0$ 时，

$$\sum_{i=1}^{n}(-1)^{i+1}C_n^i i^0=\sum_{i=1}^{n}(-1)^{i+1}C_n^i$$

$$(1-1)^n=1-\sum_{i=1}^{n}(-1)^{i+1}C_n^i=0$$

$$\sum_{i=1}^{n}(-1)^{i+1}C_n^i=1$$

为节省篇幅，这里只给出 $\sum_{i=1}^{n}(-1)^{i+1}iC_n^i=0$ （即 $k=1$ ）的归纳法证明.

（1）当 $n=2$ 时，

$$\sum_{i=1}^{2}(-1)^{i+1}iC_n^i=1\times C_2^1-2\times C_2^2=0$$

引理成立.

（2）设 $n=p-1$ 时引理成立，即 $\sum_{i=1}^{p-1}(-1)^{i+1}iC_{p-1}^i=0$，则当 $n=p$ 时，

$$\begin{aligned}\sum_{i=1}^{p}(-1)^{i+1}iC_p^i&=\sum_{i=1}^{p}(-1)^{i+1}i(C_{p-1}^i+C_{p-1}^{i-1})\\&=(-1)^{p+1}p(C_{p-1}^p+C_{p-1}^{p-1})+\sum_{i=1}^{p-1}(-1)^{i+1}i(C_{p-1}^i+C_{p-1}^{i-1})\\&=(-1)^{p+1}p+\sum_{i=1}^{p-1}(-1)^{i+1}iC_{p-1}^{i-1}\\&=(-1)^{p+1}p-\sum_{j=0}^{p-2}(-1)^{j+1}(j+1)C_{p-1}^j\\&=(-1)^{p+1}p-(-1)^1C_{p-1}^0-\sum_{j=1}^{p-2}(-1)^{j+1}(j+1)C_{p-1}^j\\&=(-1)^{p+1}p+1-\sum_{j=1}^{p-1}(-1)^{j+1}(j+1)C_{p-1}^j+(-1)^p pC_{p-1}^{p-1}\\&=0\end{aligned}$$

上面证明了 $\sum_{i=1}^{n}(-1)^{i+1}iC_n^i=0$.

使用同样的方法可以证明，对于任意 $k\,(=1,2,\cdots,n-1)$有

$$\sum_{i=1}^{n}(-1)^{i+1}i^kC_n^i=0$$

综合可知引理成立.

【定理 1】若函数 $y=f(x)$，$x\in[a,b]$，$T_n(x)$ 是 $f(x)$ 在 $x=a$ 处展开的 n 阶 Taylor 级数，$z_n(x)$ 是以 $x_0=a$，$x_i=a+ih$（$i=1,2,\cdots,n$，$h=\dfrac{b-a}{n}$）为节点的 n 阶幂级数代数插值多项式. 若 $\dfrac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{(n+1)}$ 可忽略不计，则

$$\begin{cases} y_j^{(k)}=\sum\limits_{i=k}^{n}\dfrac{f^{(i)}(x_0)}{i!}(jh)^{i-k} & j=1,2,\cdots n-k+1 \\ \alpha_k=\dfrac{f^{(k)}(x_0)}{k!} & k=1,2,\cdots,n \end{cases}$$

【证明】用归纳法证明上面的结论.

（1）当 $k=1$ 时，

$$y_j^{(1)}=\frac{y_j-\alpha_0}{jh}=\left(\sum_{i=0}^{n}\frac{f^{(i)}(x_0)}{i!}(jh)^i-f(x_0)\right)\Big/jh=\sum_{i=1}^{n}\frac{f^{(i)}(x_0)}{i!}(jh)^{i-1}$$

$$\alpha_1=\sum_{j=1}^{n}(-1)^{j+1}C_n^jy_j^{(1)}=\sum_{j=1}^{n}(-1)^{j+1}C_n^j\sum_{i=1}^{n}\frac{f^{(i)}(x_0)}{i!}(jh)^{i-1}$$

$$=\sum_{i=1}^{n}\frac{f^{(i)}(x_0)}{i!}h^{i-1}\sum_{j=1}^{n}(-1)^{j+1}C_n^jj^{i-1}=\frac{f^{(1)}(x_0)}{1!}$$

定理成立.

（2）令 $k=m-1$ 时成立，即

$$\begin{cases} y_j^{(m-1)}=\sum\limits_{i=m-1}^{n}\dfrac{f^{(i)}(x_0)}{i!}(jh)^{i-m+1} & j=1,2,\cdots n-k+1 \\ \alpha_{m-1}=\dfrac{f^{(m-1)}(x_0)}{(m-1)!} & k=1,2,\cdots,n \end{cases}$$

当 $k=m$ 时，

$$y_j^{(m)}=\frac{y_j^{(m-1)}-\alpha_{m-1}}{jh}=\sum_{i=m}^{n}\frac{f^{(i)}(x_0)}{i!}(jh)^{i-m}$$

$$\begin{aligned}\alpha_m &= \sum_{j=1}^{n-m+1}(-1)^{j+1}C_{n-m+1}^{j}y_j^{(m)}\\ &= \sum_{j=1}^{n-m+1}(-1)^{j}C_{n-m+1}^{j}\sum_{i=m}^{n}\frac{f^{(i)}(x_0)}{i!}(jh)^{i-m}\\ &= \sum_{i=m}^{n}\frac{f^{(i)}(x_0)}{i!}h^{i-m}\sum_{j=1}^{n-m+1}(-1)^{j+1}C_{n-m+1}^{j}j^{i-m}\\ &= \frac{f^{(m)}(x_0)}{m!}\end{aligned}$$

定理成立.

综上，定理 1 成立.

【定理 2】若函数 $y=f(x)$，$x\in[a,b]$，$T_n(x)$ 是 $f(x)$ 在 $x=x_0=\dfrac{a+b}{2}$ 处展开的 $2m$ 阶 Taylor 级数，且 $\dfrac{f^{(2m+1)}(\xi)}{(2m+1)!}(x-a)^{\frac{2m+1}{2}}$ 可忽略不计，$z_{2m}(x)$ 是以 $x_0=\dfrac{a+b}{2}$，$x_{2i}=x_0-\sqrt{ih}$（$i=1,2,\cdots,m$，$h=\dfrac{b-a}{n}$）为节点的 $2m$ 阶平方等距代数插值多项式，则

$$\alpha_k=\frac{f^{(k)}(x_0)}{k!}=t_k\,.$$

证明方法和过程与定理 1 的相同，证明略.

【定理 3】对于函数 $y=f(x)$，$x\in[a,b]$ 至少有一阶导数，$z_n(x)$ 是 n 阶等距节点插值多项式或 $n=2m$ 阶平方等距节点插值多项式，则

$$\lim_{h\to 0}z_n(x)=f(x)\,.$$

【证明】若 $x=x_i$（$i=0,1,\cdots,n$），即 x 为节点，显然

$$z_n(x_i)=f(x_i)$$

满足定理 3.

对于等距插值多项式，设 $x\in[x_i,x_{i+1})$，$i=0,1,\cdots,n$.

$$f(x)=f(x_i)+(x-x_i)f'(\xi_1)$$

$$z_n(x)=z_n(x_i)+(x-x_i)z_n'(\xi_2)$$

$$\left|f(x)-z_n(x)\right|=(x-x_i)\left|f'(\xi)-z_n'(x_n)\right|<h\left|f'(\xi_1)-z_n'(\xi_2)\right|$$

所以

$$\lim_{h\to 0}\left|f(x)-z_n(x)\right|=\lim_{h\to 0}h\left|f'(\xi_1)-z_n'(\xi_2)\right|=0\,,$$

即

$$\lim_{h\to 0}z_n(x)=f(x)\,.$$

上面的定理对所有插值多项式都成立.

6.4.2　代数插值多项式稳定性分析

虽然所有代数插值多项式的收敛性是相同的，但是稳定性却大不相同，原因是计算顺序不同．显然，Lagrange 插值多项式稳定性最差，Newton 插值多项式次之，幂级数型插值多项式中平方等距插值多项式最好.

1．Runge 现象

19 世纪中期 Runge 发现了 Runge 现象.

Runge 用 10 阶等距 Lagrange 插值多项式逼近连续函数 $f(x)=\dfrac{1}{1+25x^2}$，其计算结果见表 6.6.

表 6.6　10 阶 Lagrange 插值多项式逼近 $1/(1+25x^2)$

x	$\dfrac{1}{1+25x^2}$	$P_{10}(x)$	x	$\dfrac{1}{1+25x^2}$	$P_{10}(x)$
−1.00	0.03846	0.03846	−0.46	0.15898	0.24145
−0.96	0.04160	1.80438	−0.40	0.20000	0.20000
−0.90	0.04706	1.57872	−0.36	0.23585	0.18878
−0.86	0.05131	0.88808	−0.30	0.30769	0.23535
−0.80	0.05882	0.05882	−0.26	0.37175	0.31650
−0.76	0.6477	−0.20130	−0.20	0.50000	0.50000
−0.70	0.07547	−0.22620	−0.16	0.60976	0.64316
−0.66	0.08410	−0.10832	−0.10	0.80000	0.84340
−0.60	0.10000	0.10000	−0.06	0.91743	0.94090
−0.56	0.11312	0.19873	0.00	1.0000	1.0000
−0.50	0.13793	0.25376			

注意：带框者为节点值，表中只给出[−1,0]内的计算结果，且用计算机计算时人为地只取了小数点后 5 位数字.

函数 $f(x)=\dfrac{1}{1+25x^2}$（$x\in[-1,1]$）无间断点，$\dfrac{1}{26}<\dfrac{1}{1+25x^2}\leqslant 1$．但表中结果有负值，也有大于 1 的．显然不是令人满意的结果．人们把这一现象称为 Runge 现象，产生 Runge 现象的原因如下.

（1）当 $|x|<\dfrac{1}{5}$ 时，$f(x)=\lim\limits_{n\to\infty}\sum\limits_{i=0}^{n}(-5x^2)^n$，也就是 $f(x)$ 在 $x=0$ 点所展开的 Taylor 级数的收敛半径为 $\dfrac{1}{5}$，Runge 所取的区间不在收敛半径之内，不满足定理 1；

（2）Lagrange 插值多项式逼近函数时稳定性不好；

（3）Runge 时代没有计算机，手算时 Runge 取小数点后 5 位计算量够大了，但舍入误差仍不小.

实际上，若用双精度数计算该例，将插值范围由$[-1,1]$变成$[-2,2]$，用 Newton 插值多项式逼近 $f(x)$，计算 Newton 插值多项式用仿秦九韶法，虽然有一定误差，但不会出现 Runge 现象．不改变插值区间，用平方等距插值多项式逼近该函数的效果仍不错．下面附上两个程序及计算结果.

（1）10 阶 Newton 插值多项式逼近$1/(1+x^2)$的专用程序.

```
#define  N  10

void main()
{
    int i, j, m;
    double h, h0, z, s, x[N+1]={0}, c[N+1][N+1]={0};
    int duandian=2;

    h=0.2;
    for(i=0; i<=N; i++)
    {
        x[i]=-duandian+i*h;
        c[i][0]=1.0/(1+x[i]*x[i]);
    }
    for(j=1; j<=N; j++)
        for(i=0; i<N-j; i++)
            c[i][j]=(c[i+1][j-1]-c[i][j-1])/(x[i+j]-x[i]);
    h0=h/2.0;
    m=int(duandian/h0);
    for(i=1; i<m; i+=2)
    {
        z=-duandian+i*h0;
        s=c[0][N];
        for(j=N-1; j>=0; j--)
            s=s*(z-x[j])+c[0][j];
        printf("z=%-13.6lf,  s=%-13.6lf,  r=%-13.6lf\n", z, s, 1/(1+z*z));
    }
}
```

以下是程序计算结果：

```
z=-1.900000   ,  s=0.216941    ,  r=0.216920
z=-1.700000   ,  s=0.257065    ,  r=0.257069
z=-1.500000   ,  s=0.307694    ,  r=0.307692
z=-1.300000   ,  s=0.371746    ,  r=0.371747
z=-1.100000   ,  s=0.452489    ,  r=0.452489
z=-0.900000   ,  s=0.552486    ,  r=0.552486
z=-0.700000   ,  s=0.671142    ,  r=0.671141
```

```
z=-0.500000   ,  s=0.799998    ,  r=0.800000
z=-0.300000   ,  s=0.917433    ,  r=0.917431
z=-0.100000   ,  s=0.990298    ,  r=0.990099
```

（2）平方等距节点插值多项式计算的通用程序.

```
#define  N  10
#define  A  -1.0
#define  B  1.0

double f(double x)
{
    double y;
    y=1.0/(1.0+25*x*x);
    return y;
}
void main(){
    int i, j, k, p, c0, m, q, l;
    double h, h0, y0, s, z, t, c[N/2+1][N/2+1];
    double x[N+1], y[N+1], u[N+1], v[N+1], a[N+1];
    m=N/2;
    h=0.25*(B-A)*(B-A)/m;
    x[0]=(B+A)*0.5;
    y[0]=f(x[0]);
    for(i=1; i<=m; i++)
    {
        t=sqrt(i*h);
        q=2*i;
        x[q-1]=x[0]+t;
        x[q]=x[0]-t;
        y[q-1]=f(x[q-1]);
        y[q]=f(x[q]);
        u[i]=(y[q-1]-y[q])/(2*t);
        v[i]=(y[q-1]+y[q]-2*y[0])/(2*i*h);
    }
    a[0]=y[0];
    for(i=0; i<=m; i++)
        c[i][0]=c[i][i]=1;
    for(i=1; i<=m; i++)
        for(j=1; j<i; j++)
            c[i][j]=c[i-1][j]+c[i-1][j-1];
    for(k=1; k<=m; k++)
    {
        l=m-k+1;
        q=2*k;
        a[q-1]=a[q]=0.0;
        c0=1;
        for(i=1; i<=l; i++)
```

```
            {
                a[q-1]=a[q-1]+c0*c[l][i]*u[i];
                a[q]=a[q]+c0*c[l][i]*v[i];
                c0=-c0;
            }
            for(j=1; j<l; j++)
            {
                t=j*h;
                u[j]=(u[j]-a[q-1])/t;
                v[j]=(v[j]-a[q])/t;
            }
        }
        h0=0.2;
        p=(int)((B-A)/h0);
        for(i=1; i<p; i++)
        {
            z=A+i*h0;
            y0=f(z);
            s=a[N];
            for(j=N-1; j>=0; j--)
                s=s*(z-x[0])+a[j];
            printf("%-13.6f,%-13.6f, %-13.6f\n",z, y0,s);
        }
    }
```

以下是程序运行结果：

```
-0.800000    , 0.058824    , 0.054945
-0.600000    , 0.100000    , 0.093407
-0.400000    , 0.200000    , 0.230769
-0.200000    , 0.500000    , 0.699301
0.000000     , 1.000000    , 1.000000
0.200000     , 0.500000    , 0.699301
0.400000     , 0.200000    , 0.230769
0.600000     , 0.100000    , 0.093407
0.800000     , 0.058824    , 0.054945
```

第一个程序的输出结果中 z 为坐标值，s 为 Newton 插值多项式计算值，r 为理论值（的近似值）. 第二个输出结果左边是坐标值，中间是理论值，右边是计算值.

以上两个程序的运行结果说明，不是高阶代数插值多项式一定会产生 Runge 现象，另外也说明了幂级数型平方等距插值多项式的稳定性好于 Newton 插值多项式和 Lagrange 插值多项式的稳定性.

为了使算例典型，有代表性，作者请了中国科学院成都计算所原算法室徐明保副研究员、刘绍中研究员提供了若干算例，书中算例是其中的一部分.

为了节省篇幅，书中只给出 $f(x)$，n，$\sqrt{mh}$ 的值，当 $x^* \neq 0$ 时才给出 x^* 值，对于计算误差的绝对值的最大值未超过 10^{-8} 的所有计算不给出理论值. 表 6.6 中的理论值可口算出，

因此不给出.

【例 1】$y=1/(1+4x^2)$，$\sqrt{mh}=0.25$，$n=40$，计算 $P_n(x)$ 各系数，计算结果见表 6.7.

表 6.7　例 1 的计算结果

i	$A(i)$	i	$A(i)$	i	$A(i)$
0	1.0000000	14	−16370.1250800	28	90359295.6380000
1	0.0000000	15	0.0000000	29	0.0000000
2	−4.0000000	16	65259.6434580	30	−180388318.9000000
3	0.0000000	17	0.0000000	31	0.0000000
4	16.0000000	18	−257822.3598000	32	275978445.1300000
5	0.0000000	19	0.0000000	33	0.0000000
6	−63.9999956	20	994657.1138500	34	−309341768.3000000
7	0.0000000	21	0.0000000	35	0.0000000
8	255.9996823	22	−3650448.3830000	36	248290043.5800000
9	0.0000000	23	0.0000000	37	0.0000000
10	−1023.9845740	24	12287220.2240000	38	−148178592.0000000
11	0.0000000	25	0.0000000	39	0.0000000
12	4095.4645095	26	−36347025.270000000	40	60733964.6300000
13	0.0000000	27	0.0000000		

【例 2】$y=|x|$，$\sqrt{mh}=1$，$n=20$，计算 $P_n(x)$ 的值，计算结果见表 6.8.

表 6.8　例 2 的计算结果

x	$P_n(x)$	$f(x)$	x	$P_n(x)$	$f(x)$
0	0.0000000	0.0000000	0.52	0.5198958	0.5200000
0.04	0.0094448	0.0400000	0.56	0.5600332	0.5600000
0.08	0.0363679	0.0800000	0.60	0.6000504	0.6000000
0.12	0.0769234	0.1200000	0.64	0.6399887	0.6400000
0.16	0.1258671	0.1600000	0.68	0.6799724	0.6800000
0.20	0.1778946	0.2000000	0.72	0.7200143	0.7200000
0.24	0.2288480	0.2400000	0.76	0.7600169	0.7600000
0.28	0.2764242	0.2800000	0.80	0.7999710	0.8000000
0.32	0.3202148	0.3200000	0.84	0.8400074	0.8400000
0.36	0.3611608	0.3600000	0.88	0.8800530	0.8800000
0.40	0.4007147	0.4000000	0.92	0.9198321	0.9200000
0.44	0.4400802	0.4400000	0.96	0.9603561	0.9600000
0.48	0.4798179	0.4800000	1	1.0000000	1.0000000

【例 3】$y=\sin^2 x/x$，$\sqrt{mh}=0.5$，$n=20$，计算 $P_n(x)$ 的 0～1 阶导数值，计算结果见表 6.9.

表 6.9 例 3 的计算结果

x	$P_n(x)$		$P_n'(x)$	$f'(x)$	x	$P_n(x)$		$P_n'(x)$	$f'(x)$
0	0.0000000	0.0000000	1.0000000	0.0000000	0.26	0.2598020	0.2541939	0.9992385	0.9961934
0.02	0.0000000	0.0199973	1.0000000	0.9999999	0.28	0.2797132	0.2727587	0.9989759	0.9948807
0.04	0.0400000	0.0399787	0.9999996	0.9999979	0.30	0.2995952	0.2911073	0.9986505	0.9932549
0.06	0.0599999	0.0599280	0.9999978	0.9999892	0.32	0.3194410	0.3092254	0.9982533	0.9912701
0.08	0.0799995	0.0798295	0.9999932	0.9999659	0.34	0.3392432	0.3270989	0.9977743	0.9988773
0.10	0.0999983	0.0996671	0.9999833	0.9999167	0.36	0.3589931	0.3447143	0.9972030	0.9860243
0.12	0.1199959	0.1194251	0.9999654	0.9998272	0.38	0.3786808	0.3620579	0.9965284	0.9826565
0.14	0.1399910	0.1390877	0.9999360	0.9996799	0.40	0.3982955	0.3791166	0.9957388	0.9787158
0.16	0.1599825	0.1586393	0.9998908	0.9994539	0.42	0.4178252	0.3958776	0.9948219	0.9741417
0.18	0.1799685	0.1780644	0.9998250	0.9991253	0.44	0.4372565	0.4123282	0.9937649	0.9688711
0.20	0.1999467	0.1973475	0.9997334	0.9986669	0.46	0.4565750	0.4284564	0.9925543	0.9628380
0.22	0.2199141	0.2164735	0.9996096	0.9980483	0.48	0.4757645	0.4442500	0.9911761	0.9559742
0.24	0.2398673	0.2354272	0.9994471	0.9972360	0.50	0.4948079	0.4596977	0.9896158	0.9482090

【例 4】$y=\sqrt{1+x}$，$\sqrt{mh}=0.5$，$n=20$，$x^*=1$，计算 $P_n(x)$ 的 0～5 阶导数值，计算结果见表 6.10.

表 6.10 例 4 的计算结果

x	$P(x)$	$P'(x)$	$P''(x)$	$P'''(x)$	$P^{(4)}(x)$	$P^{(5)}(x)$
1	1.4142136	0.3535534	−0.0883883	0.0662913	−0.0828641	0.1450122
1.02	1.4282857	0.3517988	−0.0870789	0.0646626	−0.0800279	0.1386622
1.04	1.4422205	0.3500700	−0.0858015	0.0630893	−0.0773153	0.1326489
1.06	1.4560220	0.3483665	−0.0845550	0.0615692	−0.0747199	0.1269512
1.08	1.4696938	0.3466876	−0.0833384	0.0600998	−0.0722353	0.1215498
1.10	1.4832397	0.3450328	−0.0821507	0.0586790	−0.0698560	0.1164267
1.12	1.4966630	0.3434014	−0.0809909	0.0573049	−0.0675756	0.1115650
1.14	1.5099669	0.3417930	−0.0798582	0.559784	−0.0653918	0.1069461
1.16	1.5231546	0.3402069	−0.0787516	0.546886	−0.0632970	0.1025646
1.18	1.5392291	0.3386427	−0.0776704	0.0534429	−0.0612877	0.0983977
1.20	1.5491933	0.3370999	−0.0766136	0.0522366	−0.0593597	0.0944359
1.22	1.5620499	0.3355780	−0.075806	0.0510680	−0.0575090	0.0906674
1.24	1.5748016	0.3340766	−0.0745707	0.0499357	−0.0557318	0.0870810
1.26	1.5874508	0.3325951	−0.0735830	0.0488383	−0.0540246	0.0836665
1.28	1.600000	0.3311331	−0.0726169	0.0477743	−0.0523841	0.0804142
1.30	1.6124515	0.3296902	−0.0716718	0.0467425	−0.0508070	0.0773151
1.32	1.6248077	0.3282661	−0.0707470	0.0457416	−0.0492905	0.0743607
1.34	1.6370705	0.3268602	−0.0698419	0.0447705	−0.0478317	0.0715431

（续表）

x	$P(x)$	$P'(x)$	$P''(x)$	$P'''(x)$	$P^{(4)}(x)$	$P^{(5)}(x)$
1.36	1.6492422	0.3254723	−0.0689560	0.0438280	−0.0464279	0.0688550
1.38	1.6613248	0.3241019	−0.0680886	0.0429130	−0.0452767	0.0662892
1.40	1.6733200	0.3227486	−0.0672393	0.0420246	−0.0437756	0.0638394
1.42	1.6852299	0.3214122	−0.0664075	0.0411617	−0.0425224	0.0614993
1.44	1.6970563	0.3200922	−0.0655927	0.0403234	−0.0413149	0.0592632
1.46	1.7088007	0.3187884	−0.0647944	0.0395088	−0.0401512	0.0571257
1.48	1.7204650	0.3175003	−0.0640122	0.0387170	−0.0390293	0.0550816
1.50	1.7320508	0.3162278	−0.0632456	0.0379473	−0.0379473	0.0531263

2．结果分析

对于例 1，当$|x|<0$时，$y=1/(1+4x^2)=\sum_{i=0}^{\infty}(-4x^2)^i$，奇次项系数为 0，偶次项系数为 1, -4, 16,⋯，表 6.6 的结果与之一致．偶次项系数阶愈低，与之差别愈小，高次项系数和理论值有差异的原因系计算误差引起．本文各例均采用双精度数，$\varepsilon=0.5\times10^{-16}$，$a_j$ 的计算误差为 $O(\varepsilon/h^{j/2})$，此例中 $h=0.003125$．

对于例 2，$x=0$ 处仅保证函数连续，$n=20$ 也未出现 Runge 现象．

例 3 的函数在 $x=0$ 处只有一阶导数，函数逼近的最大误差为10^{-15}．导数逼近为10^{-3}，也未产生 Runge 现象，x=0 处导数误差例外．

例 4 特别选择 $x^*=1\neq0$，插值范围在 Taylor 级数收敛半径之内，因此选了 0～5 阶导数逼近，5 阶导数逼近结果误差未超过10^{-4}．

由于 Runge 的误导，近百年来研究高阶代数插值者甚少，代数插值多项式的许多良好性质还鲜为人知，例如，当函数连续性有限时截断误差的估计，Weierstass 定理 $P_n(x)$ 和 Taylor 级数之间的关系，无限和有限项和之间的联系等，这些问题亟待研究．

习题 6

1．已知函数 $f(x)$ 的观测数据为 $f(1)=1$，$f(4)=2$，$f(2)=1$，求以 1, 4, 2 为节点的 Lagrange 插值多项式，并求 $f(1.5)$ 的近似值．

2．试利用 100, 121 和 144 的平方根求 $\sqrt{115}$ 的近似值．

3．设 x_0，x_1,⋯，x_n 为 $n+1$ 个相异的插值节点，$l_i(x)$（$i=0,1,\cdots,n$）为 Lagrange 插值多项式，证明

（1）$\sum_{i=0}^{n}l_i(x)=1$；

（2）$\sum_{i=0}^{n}x_i^j l_i(x)=x^j$，$j=1,2,\cdots,n$；

（3）$\sum_{i=0}^{n} x_i^j l_i(0) = \begin{cases} 1 & j=0 \\ 0 & j=1,2,\cdots,n \\ (-1)^n x_0 x_1 \cdots x_n & j=n+1 \end{cases}$.

4．设 $f(x)$ 为 x 的 n 次多项式，证明：当 $k>n$ 时，$f[x_0,x_1,\cdots,x_k]=0$.

5．已知 $f(x)$ 的函数值 $f(0)=-5$，$f(1)=-3$，$f(-1)=-15$，$f(2)=-9$，求次数不高于三次的 Newton 插值多项式 $N(x)$ 及 $f(1.5)$ 的近似值.

6．求次数不高于四次的 Newton 插值多项式 $N(x)$，满足条件

$$N(0)=N'(0)=1，\ N(1)=N'(1)=2，\ N(3)=3.$$

7．求次数不高于三次的 Newton 插值多项式 $N(x)$，满足条件

$$N(1)=2，\ N'(1)=3，\ N''(1)=2，\ N(2)=5.$$

8．求次数不高于三次的新代数插值多项式 $Z(x)$，满足条件

$$Z(1)=-3，\ Z(0)=-5，\ Z(-1)=15，\ Z(2)=-9.$$

9．求次数不高于四次的新代数插值多项式 $Z(x)$，满足条件

$$Z(-2)=1，\ Z(-1)=2，\ Z(0)=3，\ Z(1)=5，\ Z(2)=6.$$

第7章　样 条 函 数

自 19 世纪中期 I. J. Schoenberg 把样条函数引入数学中，样条函数就成为现代数学中的一个十分活跃的分支．样条函数起源于船舶外形设计，我国人民大会堂的屋顶就利用了样条函数才保证了如期完工．

样条函数分为 B 样条和一般样条两类，本书只介绍一般样条函数，样条函数简称样条．

样条函数实质上就是分段低阶插值多项式，并对各段公共点做了一些特殊处理，对于一般样条函数，用得最广的是二次样条和三次样条．

7.1　二次样条函数

二次样条函数是分段的带插值函数特性的多项式．

7.1.1　二次样条函数特性

设 x_i（$i=0,1,\cdots,n$）为节点，二次样条函数是分段二次多项式，可表述成

$$S(x)=\begin{cases} S_0(x), & x_0 \leqslant x < x_1 \\ S_1(x), & x_1 \leqslant x < x_2 \\ \cdots & \cdots \\ S_{n-1}(x), & x_{n-1} \leqslant x \leqslant x_n \end{cases} \tag{7.1-1}$$

上式中，

$$S_i(x)=a_i(x-x_{i+1})(x-x_i)+b_i(x-x_i)+c_i\ ,\ \ x_{i-1} \leqslant x < x_i\ ,\ \ i=0,1,\cdots,n-1\ .$$

二次样条函数有以下特性：

（1）二次样条函数是分段带插值函数特性的二次多项式，满足

$$\begin{cases} S_i(x_i)=f(x_i)=y_i, & i=0,1,\cdots,n-1 \\ S_{n-1}(x_n)=f(x_n)=y_n \end{cases} \tag{7.1-2}$$

（2）二次样条函数本身是连续函数，满足

$$S_i(x_{i+1})=S_{i+1}(x_{i+1})\ ,\ \ i=0,1,\cdots,n-2 \tag{7.1-3}$$

（3）二次样条函数是一阶导数连续的函数，满足

$$S_i'(x_{i+1})=S_{i+1}'(x_{i+1})\ ,\ \ i=0,1,\cdots,n-2 \tag{7.1-4}$$

7.1.2　二次样条函数系数确定

由式（7.1-1）和式（7.1-2）有

$$S_i(x_i)=a_i(x_i-x_{i+1})(x_i-x_i)+b_i(x_i-x_i)+c_i=y_i$$
$$c_i=y_i,\ \ i=0,1,\cdots,n-1 \tag{7.1-5}$$

由式（7.1-1）和式（7.1-3）有

$$b_i=\frac{y_{i+1}-y_i}{x_{i+1}-x_i},\ \ i=0,1,\cdots,n-2$$

由 $S_{n-1}(x_n)=b_{n-1}(x_n-x_{n-1})+c_{n-1}=y_n$ 得

$$b_{n-1}=\frac{y_n-c_{n-1}}{x_n-x_{n-1}}=\frac{y_n-y_{n-1}}{x_n-x_{n-1}}$$

令 $h_i=x_{i+1}-x_i$，综上得

$$b_i=\frac{y_{i+1}-y_i}{x_{i+1}-x_i}=\frac{y_{i+1}-y_i}{h_i},\ \ i=0,1,\cdots,n-1 \tag{7.1-6}$$

式（7.1-1）有 $3n$ 个系数，由 3 个性质可给出 $3n-1$ 个方程，还差一个条件，通常让一个边界点的导数值已知，不妨将这个点的编号认为是 0，即 $f'(x_0)=y_0'$ 已知，由此可得到

$$a_0(x_0-x_1)+b_0=y_0'$$
$$a_0=\frac{y_0'-b_0}{x_0-x_1} \tag{7.1-7}$$

由式（7.1-7）和式（7.1-4）有

$$a_{i-1}(x_i-x_{i-1})+b_{i-1}=a_i(x_i-x_{i+1})+b_i$$
$$a_i=-\frac{h_{i-1}}{h_i}a_{i-1}+\frac{b_i-b_{i-1}}{h_i},\ \ i=1,\cdots,n-1 \tag{7.1-8}$$

下面是已知 y_0' 时二次样条函数的综合计算公式：

$$\begin{cases} c_i=y_i & \\ b_i=\dfrac{y_{i+1}-y_i}{h_i} & i=0,1,\cdots,n-1 \\ a_0=\dfrac{y_0'-b_0}{h_0} & \\ a_i=-\dfrac{h_{i-1}}{h_i}a_{i-1}+\dfrac{b_i-b_{i-1}}{h_i} & i=1,2,\cdots,n-1 \end{cases} \tag{7.1-9}$$

若节点间距相等，则等距二次样条系数计算公式为

$$\begin{cases} c_i=y_i & \\ b_i=\dfrac{y_{i+1}-y_i}{h} & i=0,1,\cdots,n-1 \\ a_0=\dfrac{y_0'-b_0}{h} & \\ a_i=-a_{i-1}+\dfrac{b_i-b_{i-1}}{h} & i=1,2,\cdots,n-1 \end{cases} \tag{7.1-10}$$

7.1.3　二次样条插值计算过程和计算实例

1．计算过程

（1）按式（7.1-9）或式（7.1-10）生成二次样条函数；

（2）对每一个用二次样条函数逼近的点 z，确定其所在函数段 i，计算公式为

$$\begin{cases} i \Leftarrow 0 \\ i \Leftarrow i+1, \qquad z > x_{i+1} \end{cases}$$

当 z 大于 x_i 而不大于 x_{i+1} 时，确定了段数 i；

（3）由式（7.1-1）计算 $S_i(z)$ 的值.

2．计算实例

【例 1】 作二次样条函数以逼近 $\sqrt{x}$，节点取 $x_0=\dfrac{1}{4}$，$x_1=\dfrac{5}{8}$，$x_2=1$，并计算 $S(0.875)$ 的值.

【解】 本例节点是等距的，由题设可得 $h=\dfrac{3}{8}=0.375$，设

$$S(x)=\begin{cases} a_0\left(x-\dfrac{1}{4}\right)\left(x-\dfrac{5}{8}\right)+b_0\left(x-\dfrac{1}{4}\right)+c_0, & \dfrac{1}{4}\leqslant x<\dfrac{5}{8} \\ a_1\left(x-\dfrac{5}{8}\right)(x-1)+b_1\left(x-\dfrac{5}{8}\right)+c_1, & \dfrac{5}{8}\leqslant x\leqslant 1 \end{cases}$$

按式（7.1-10）有

（1）$c_i=y_i$，$i=0,1$

$$c_0=\sqrt{x_0}=0.5$$

$$c_1=\sqrt{5/8}=0.790569$$

（2）$b_i=\dfrac{y_{i+1}-y_i}{h}$，$i=0,1$

$$b_0=\frac{y_1-y_0}{h}=\frac{\sqrt{5/8}-0.5}{0.375}=0.774852$$

$$b_1=\frac{y_2-y_1}{h}=\frac{1-0.790569}{0.375}=0.558482$$

$$a_0=\frac{b_0-y_0'}{h}=\frac{0.774852-1}{0.375}=-0.600395$$

$$a_1=-a_0+\frac{b_1-b_0}{h}=0.600395+\frac{0.558482-0.774852}{0.375}=0.023408$$

由此有

$$S(x)=\begin{cases}-0.600395(x-0.25)(x-0.625)+0.774852(x-0.25)+0.5, & \frac{1}{4}\leqslant x<\frac{5}{8}\\ 0.023408(x-0.625)(x-1)+0.558482(x-0.625)+0.790569, & \frac{5}{8}\leqslant x\leqslant 1\end{cases}$$

当 $x=0.875$ 时，由于 0.875>0.625，所以 $i=1$，即

$$\begin{aligned}S(0.875)&=0.023408\times(0.875-0.625)\times(0.875-1)+0.558482\times(0.875-0.625)+0.790569\\&=0.929458\end{aligned}$$

7.1.4 二次样条插值余项

有了二次样条函数 $S(x)$，自然关心 $f(x)-S(x)$ 等于多少?.

【定理】设 $f(x)$ 在整个插值区间至少存在三阶连续导数，$h=\max\limits_i|h_i|$，令 $R(x)=f(x)-S(x)$，$M=\max|f'''(x)|$，则

$$R''(x)\leqslant hM$$

$$R'(x)\leqslant\frac{M}{8}h^2$$

$$R(x)\leqslant\frac{M}{8}(b-a)h^2$$

【证明】由二次样条函数的定义和性质有

$$R'(x_i)=f'(x_i)-S'(x_i)=0\ ,$$

$$R'(x_{i+1})=f'(x_{i+1})-S'(x_{i+1})=0\ ,$$

即

$$R'(x_i)=R'(x_{i+1})=0\ .$$

由罗尔定理，存在 $\xi_i\in[x_i,x_{i+1})$，使得

$$R''(\xi_i)=0\ ,$$

而

$$R''(x)=R''(\xi_i)+\int_{\xi_i}^{x}R'''(t)\mathrm{d}t=\int_{\xi_i}^{x}f'''(t)\mathrm{d}t\leqslant h_iM\leqslant hM\ .$$

由于 $R'(x_i)=R'(x_{i+1})=0$，并且 $S'(x)$ 是 $x\in[x_i,x_{i+1}]$ 上的线性插值函数，所以

$$R(x)=\frac{(x-x_1)(x-x_{i+1})}{2}f'''(\xi)\ ,\quad x_i<\xi<x_{i+1}\ .$$

由于

$$\max_{x_i\leqslant x\leqslant x_{i+1}}(x-x_i)(x-x_{i+1})=\frac{1}{4}h_i^2\ ,$$

所以

$$|R'(x)|\leqslant\frac{M}{8}h^2\ .$$

最后，考虑到

$$R(x_0)=f(x_0)-S(x_0)=0\,,$$

$$R(x)=\int_{x_0}^{x}R'(x)\mathrm{d}x\,,$$

即可由式（7.1-11）得到

$$R(x)=\frac{M}{8}(b-a)h^2\,.$$

7.2　三次样条函数

三次样条函数的力学意义清楚，因此在工程物理中用得较多．三次样条函数是分段三次多项式，鉴于其力学意义清楚，构造它时采用另一途径.

7.2.1　三次样条函数的定义

设在$[a,b]$上给出了$n+1$个互异节点

$$a=x_0<x_1<\cdots<x_n=b\,,$$

函数$f(x)$在这些节点上的值为

$$f(x_i)=y_i\,,\quad i=0,1,\cdots,n\,.$$

若函数$S(x)$满足下列条件，则称$S(x)$为函数$f(x)$关于x_i($i=0,1,\cdots,n$)的三次样条函数，简称三次样条：

（1）$S(x)$在每个子区间$[x_{i-1},x_i]$上都是不高于三次的多项式；

（2）$S''(x)$在整个区间上连续，自然函数和一阶导数也连续；

（3）$S(x_i)=f(x_i)=y_i$，$i=0,1,\cdots,n$.

n个三次多项式共有$4n$个系数，由条件（2）和（3）可提供$4n-2$个方程，不足以唯一确定所有系数，因此与确定二次样条函数一样，还须增设边界条件.

7.2.2　边界条件和边界条件类型

假设

$$S(x)=\begin{cases}S_1(x), & x\in[x_0,x_1]\\ S_2(x), & x\in[x_1,x_2]\\ \cdots & \cdots\\ S_i(x), & x\in[x_i,x_{i+1}]\\ \cdots & \cdots\\ S_{n-1}(x), & x\in[x_{n-1},x_n]\end{cases}\qquad(7.2\text{-}1)$$

式中，

$$S_i(x)=a_ix^3+b_ix^2+c_ix+d_i,\quad i=0,1,\cdots,n-1.$$

由性质（2）和性质（3）有

$$\begin{cases} S_i(x_i)=f(x_i)=y_i, & i=0,1,\cdots,n-1\\ S_{n-1}(x_n)=f(x_n)=y_n & \\ S_i(x_{i+1})=S_{i+1}(x_{i+1}) & i=0,1,\cdots,n-2\\ S_i'(x_{i+1})=S_{i+1}'(x_{i+1}), & i=0,1,\cdots,n-2\\ S_i''(x_{i+1})=S_{i+1}''(x_{i+1}), & i=0,1,\cdots,n-2 \end{cases} \tag{7.2-2}$$

式（7.2-1）中有 $4n$ 个未知数，而式（7.2-2）中只能立 $4n-2$ 个方程或等式，还差两个条件，不能唯一确定式（7.2-1）中的 $4n$ 个系数，因此必须增加两个条件．这些条件通常出自边界，故称为边界条件.

边界条件一般应视实际问题而定，其类型较多，常见的类型为以下三类.

（1）给出两个端点的一阶导数

$$S'(x_0)=y_0',\quad S'(x_n)=y_n'.$$

（2）给出两个端点的二阶导数

$$S''(x_0)=y_0'',\quad S''(x_n)=y_n''.$$

（3）当 $y=f(x)$ 是以 $T=b-a$ 为周期的函数时，边界条件为

$$S'(x_0)=S'(x_n),\quad S''(x_0)=S''(x_n).$$

注意对于周期函数，由于已利用 $S(x_0)=y_0$ 和 $S(x_n)=y_n$，故不能再有 $S(x_0)=S(x_n)$.

7.2.3 三次样条函数的构造方法

假设 $S''(x_i)=M_i$（$i=0,1,\cdots,n$），由于 $S_i(x)$ 是三次多项式，所以 $S_i''(x)$ 是一次多项式，故可设

$$S_i''(x)=\frac{x-x_i}{x_{i-1}-x_i}M_{i-1}+\frac{x-x_{i-1}}{x_i-x_{i-1}}M_i,\quad x\in[x_{i-1},x_i].$$

记 $h_i=x_i-x_{i-1}$，则

$$S_i''(x)=\frac{x-x_i}{h_i}M_{i-1}+\frac{x-x_{i-1}}{h_i}M_i,\quad x\in[x_{i-1},x_i].$$

连续积分两次有

$$\begin{cases} S_i(x)=\dfrac{(x_i-x)^3}{6h_i}M_{i-1}+\dfrac{(x-x_{i-1})^3}{6h_i}M_i+A_i(x-x_i)+B_i\\ x\in[x_{i-1},x_i],\ \ i=1,2,\cdots,n \end{cases} \tag{7.2-3}$$

利用插值条件

$$S_i(x_i)=y_i,\quad S_i(x_{i-1})=y_{i-1}$$

有

$$\begin{cases} A_i = \dfrac{y_i - y_{i-1}}{h_i} - \dfrac{h_i}{6}(M_i - M_{i-1}) \\ B_i = y_{i-1} - \dfrac{1}{6}M_{i-1}h_i^2 \end{cases}$$

这样就把三次样条的 $4n$ 个系数的问题转换成求 $n+1$ 个未知数 M_i（$i=0,1,\cdots,n$）的问题.

上面利用了插值函数的性质，现在利用样条函数的性质：

$$S_i'(x_i) = S_{i+1}'(x_i)\ ,\quad i=0,1,\cdots,n-1\ .$$

由此有

$$\begin{aligned} S_i'(x_i) &= \frac{y_i - y_{i-1}}{h_i} + \frac{y_i}{6}M_{i-1} + \frac{h_i}{3}M_i \\ &= \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{3} - \frac{h_{i+1}}{6}M_{i+1} \\ &= S_{i+1}'(x_i) \end{aligned}$$

整理后得

$$\frac{h_i}{h_i + h_{i+1}}M_{i-1} + 2M_i + \frac{h_{i+1}}{h_i + h_{i+1}}M_{i+1} = \frac{6}{h_i + h_{i+1}}\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right),\quad i=1,2,\cdots,n-1\ .$$

若记

$$\begin{cases} \mu_i = \dfrac{h_i}{h_i + h_{i+1}} \\ \lambda_i = \dfrac{h_{i+1}}{h_i + h_{i+1}} = 1 - \mu_i \\ g_i = \dfrac{6}{h_i + h_{i+1}}\left(\dfrac{y_{i+1} - y_i}{h_{i+1}} - \dfrac{y_i - y_{i-1}}{h_i}\right) \end{cases} \tag{7.2-4}$$

可得

$$\begin{cases} \mu_1 M_0 + 2M_1 + \lambda_1 M_2 = g_1 \\ \mu_2 M_1 + 2M_2 + \lambda_2 M_3 = g_2 \\ \qquad\cdots \\ \mu_{n-1}M_{n-2} + 2M_{n-1} + \lambda_{n-1}M_n = g_{n-1} \end{cases} \tag{7.2-5}$$

式（7.2-5）中共有 $n+1$ 个未知数，只有 $n-1$ 个方程，因此是一个不定方程组．要想使方程组有唯一解，必须利用边界条件，增加两个约束.

对于第一类边界条件，由

$$S_0'(x) = \frac{-(x_1 - x_0)^2}{2h_i}M_0 + \frac{(x - x_0)^2}{2h_i}M_1 - \frac{y_i - y_0}{h_i} - \frac{h_1}{6}(M_1 - M_0)\ ,$$

当

$$S'(x_0) = y_0'$$

有

$$2M_0 + M_1 = \frac{6}{h_1}\left(\frac{y_1 - y_0}{h_1} - y_0'\right) \tag{7.2-6}$$

同样，由 $S_n'(x_n) = S'(x_n) = y_n'$ 得

$$M_{n-1} + 2M_n = \frac{6}{h_n}\left(y_n' - \frac{y_n - y_{n-1}}{h_n}\right) \tag{7.2-7}$$

将方程组（7.2-5）、（7.2-6）和（7.2-7）结合就得三对角方程组

$$\begin{bmatrix} 2 & 1 & & & & \\ \mu_1 & 2 & \lambda_1 & & & \\ & \mu_2 & 2 & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \\ & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix} \tag{7.2-8}$$

其中，

$$\begin{cases} g_0 = \dfrac{6}{h_1}\left(\dfrac{y_1 - y_0}{h_1} - y_0'\right) \\ g_n = \dfrac{6}{h_n}\left(y_n' - \dfrac{y_n - y_{n-1}}{h_n}\right) \end{cases}$$

在第二类边界条件下，由于 $M_0 = y_0''$， $M_n = y_n''$，将 M_0 和 M_n 代入式（7.2-4）可以得到

$$\begin{bmatrix} 2 & \lambda_1 & & & & \\ u_2 & 2 & \lambda_2 & & & \\ & \mu_3 & 2 & \lambda_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & & \mu_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} g_1 - u_1 y_0'' \\ g_2 \\ g_3 \\ \vdots \\ g_{n-2} \\ g_{n-1} - \lambda_{n-1} y_n'' \end{bmatrix} \tag{7.2-9}$$

对于第三类边界条件，由周期函数性质显然有

$$M_0 = M_n，\quad y_0 = y_n，$$

由 $S'(x_0) = S'(x_n)$ 有

$$-\frac{h_1}{2}M_0 + \frac{y_1 - y_0}{h_1} - \frac{h_1}{6}(M_1 - M_0) = \frac{h_n}{2}M_n + \frac{y_n - y_{n-1}}{h_n} - \frac{h_n}{6}(M_n - M_{n-1})，$$

将 $M_0 = M_n$， $y_0 = y_n$ 代入上式得

$$\frac{h_1}{h_1 + h_n}M_1 + \frac{h_n}{h_1 + h_n}M_{n-1} + 2M_n = \frac{6}{h_1 + h_n}\left(\frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}\right).$$

将 $M_0 = M_n$ 代入式（7.2-5）中第一个方程得

$$2M_1 + \lambda_1 M_2 + \mu_1 M_n = g_1$$

由此有

$$\begin{bmatrix} 2 & \lambda_1 & & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & & \\ & \mu_3 & 2 & \lambda_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix} \tag{7.2-10}$$

很容易证明上述三个方程组的系数行列式都不为 0，三个方程组都有唯一解.

当 $h_0 = h_1 = \cdots = h_{n-1} = h$ 时，即节点等距时，有

$$\begin{cases} \mu_i = \lambda_i = \dfrac{1}{2} \\ g_i = \dfrac{3(y_{i+1} + y_i)}{h^2} \end{cases}, \quad i = 0,1,\cdots,n-1 \tag{7.2-11}$$

此时三个方程组都可大大简化.

M_i 相应于材料力学或弹性力学中樑的弯矩，而每一个方程中只出现相邻的三个弯矩，力学中称这类方程组为三弯矩方程组，求这些弯矩的方法称为三弯矩方法. 三次样条函数的构造方法及所得到的方程组的系数和右端项，与力学中的三弯矩方程组完全一样，只不过物理意义不同. 实际上，有一些不同的问题可能会归结于同一数学问题，例如受正压的薄膜和樑的扭转，最后都会解一完全类似的偏微分方程.

三次样条函数具有不少良好的性质，因篇幅关系此处不予介绍.

7.2.4　三次样条计算过程

三次样条函数随边界条件的不同，其计算过程略有不同.

1. 第一类边界条件三次样条函数计算过程

（1）计算 h_i（$i = 0,1,\cdots,n$）：

$$h_i = x_i - x_{i-1}$$

（2）计算 u_i，λ_i 和 g_i：

$$\begin{cases} \mu_i = \dfrac{h_i}{h_i + h_{i+1}} \\ \lambda_i = 1 - \mu_i \\ g_i = \dfrac{6}{h_i + h_{i+1}} \left(\dfrac{y_{i+1} - y_i}{h_{i+1}} - \dfrac{y_i - y_{i-1}}{h_i} \right) \\ g_0 = \dfrac{6}{h_i} \left(\dfrac{y_1 - y_0}{h_i} - y_0' \right) \\ g_n = \dfrac{6}{h_n} \left(y_n' - \dfrac{y_n - y_{n-1}}{h_n} \right) \end{cases}$$

（3）生成三对角方程组.

（4）给出待插值计算点的值 z，计算 z 所属三次样条函数的函数段 i，计算公式为

$$\begin{cases} i=0 \\ i \Leftarrow i+1 \qquad z > x_{i+1} \end{cases}$$

（5）按式（7.2-3）计算 $S_i(z)$.

2．第二类边界条件三次样条函数计算过程

（1）计算 h_i（$i=1,2,\cdots,n$）：

$$h_i = x_i - x_{i-1}$$

（2）计算 μ_i，λ_i 和 g_i：

$$\begin{cases} \mu_i = \dfrac{h_i}{h_i + h_{i+1}} \\ \lambda_i = 1 - \mu_i & i=1,2,\cdots,n-1 \\ g_i = \dfrac{6}{h_i + h_{i+1}}\left(\dfrac{y_{i+1}-y_i}{h_{i+1}} - \dfrac{y_i - y_{i-1}}{h_i}\right) \end{cases}$$

（3）按式（7.2-9）生成方程组

$$\begin{bmatrix} 2 & \lambda_1 & & & & \\ \mu_2 & 2 & \lambda_2 & & & \\ & \mu_3 & 2 & \lambda_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & & \mu_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} g_1 - \mu_1 y_0'' \\ g_2 \\ g_3 \\ \vdots \\ g_{n-2} \\ g_{n-1} - \lambda_{n-1} y_n'' \end{bmatrix}$$

并解方程组，算出 M_i（$i=1,2,\cdots,n$）.

（4）给出待插值点 z，用下式计算 z 所属区间 i：

$$\begin{cases} i=0 \\ i \Leftarrow i+1, \qquad z > x_{i+1} \end{cases}$$

（5）按下式计算 $S(z)$ 值：

$$\begin{cases} S(z) = \dfrac{(x_i - x)^3}{6h_i} M_{i-1} + \dfrac{(x - x_{i-1})^3}{6h_i} M_i + A_i(x - x_i) + B_i \\ A_i = \dfrac{y_i - y_{i-1}}{h_i} - \dfrac{h_i}{6}\left(M_i - M_{i-1}\right) \\ M_0 = y_0'' \\ M_n = y_n'' \end{cases}$$

3．第三类边界条件三次样条计算过程

略.

7.2.5　三次样条计算实例

【例 1】给定下列各值：

x	0	1	2	3
y	−1	−1	0	0

求满足边界条件 $S'(0)=1$，$S'(3)=2$ 的三次样条函数，并计算 $S(0.5)$ 和 $S(1.5)$ 的值.

【解】本例是计算等距三次样条函数，并做插值计算，求解过程如下.

（1）计算 h，$h=h_1=h_2=h_3=1$；

（2）计算 μ_i，λ_i 和 g_i：

$\mu_1=\mu_2=0.5$

$\lambda_1=\lambda_2=0.5$

$g_0=-6$，$g_1=3$，$g_2=3$，$g_3=0$

（3）由（1）和（2）得方程组

$$\begin{bmatrix} 2 & 1 & 0 & 0 \\ 0.5 & 2 & 0.5 & 0 \\ 0 & 0.5 & 2 & 0.5 \\ 0 & 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{bmatrix}=\begin{bmatrix} -6 \\ 3 \\ 3 \\ 0 \end{bmatrix}$$

解之得

$$M_0=-\frac{62}{15},\quad M_1=\frac{102}{45},\quad M_2=\frac{48}{45},\quad M_3=-\frac{24}{45}.$$

（4）综上得三次样条函数

$$S(x)=\begin{cases} \dfrac{-31(1-x)^3}{45}+\dfrac{17x^3}{45}-\dfrac{14(1-x)}{45}-\dfrac{8}{15}, & x\in[0,1] \\ \dfrac{17(2-x)^3}{45}+\dfrac{8(x-1)^3}{45}-\dfrac{62(2-x)}{45}-\dfrac{8(x-1)}{45}, & x\in[1,2] \\ \dfrac{8(3-x)^3}{45}-\dfrac{4(x-2)^3}{45}-\dfrac{8(3-x)}{45}+\dfrac{4(x-2)}{45}, & x\in[2,3] \end{cases}$$

$x=0.5\in[0,1]$，$i=0$，所以

$$S(0.5)=-31(1-0.5)^3+\frac{17\times0.5^3}{45}-\frac{14(1-0.5)}{45}-\frac{8}{15}=0.883333.$$

$x=1.5\in[1,2]$，$i=1$，所以

$$S(1.5)=\frac{17\times(2-1.5)^3}{45}+\frac{8\times(1.5-1)^3}{45}-\frac{62\times(2-1.5)}{45}-\frac{8\times(1.5-1)}{45}=-0.708333.$$

习题 7

1．下面是一分段多项式：

$$y=\begin{cases}2x^2+4.5, & 0\leqslant x<2\\ x^2+4x+0.5, & 2\leqslant x<3\\ 2x^2-2x+9.5, & 3\leqslant x<100\end{cases}$$

该多项式是否可作为二次样条函数?

2．下面是一分段多项式：

$$y=\begin{cases}9x-3, & 0\leqslant x<2\\ 2x^2+x+5, & 2\leqslant x<3\\ 3x^2-5x+14, & 3\leqslant x\leq 5\end{cases}$$

该多项式是否可作为二次样条函数?

3．作一个二次样条函数 $s(x)$，满足条件

$$s(0)=5,\quad s(1)=7,\quad s(2)=3,\quad s'(2)=4.$$

4．作一个二次样条函数 $s(x)$，满足条件

$$s(1)=2,\quad s(2)=2,\quad s(3)=3,\quad s(4)=5,\quad s'(1)=2.$$

5．判断分段三次多项式

$$y=\begin{cases}x^3+2x^2+5x+7, & -1\leqslant x<0\\ 2x^3+x^2+5x+7, & 0\leqslant x<1\\ 4x^3-10x^2+23x-1, & 1\leqslant x<2\end{cases}$$

是否可作为三次样条函数.

6．作一个三次样条函数 $s(x)$，满足条件

$$s(1)=1,\quad s(2)=2,\quad s(3)=2,\quad s(4)=3,\quad s'(1)=2,\quad s'(4)=4.$$

第8章　有 理 插 值

代数插值多项式和样条函数，是逼近光滑函数的最简单的也是较为有效的工具．但是对于有理函数和有奇点的函数及某些点无界的函数，逼近效果常常不佳．若用以逼近这些函数的函数本身在这些点也是奇点，可能效果会好些．有理函数也是一种较简单的函数，而且在某些点也是无界的，将某些在有些点无界的函数用它做逼近工具，效果会相当不错．

有理式都可写成连分式．

8.1　连分式

数学算式中有一种特殊算式——连分式，有理函数都可用连分式表示．

8.1.1　连分式概念

【定义 1】形如

$$R_n = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \ddots + \cfrac{}{+b_{n-1} + \cfrac{a_n}{b_n}}}} \tag{8.1-1}$$

的算式称为连分式．式中 b_n 为常数，$a_i\,(i=1,2,\cdots,n)$为常数，$b_i\,(i=0,1,\cdots,n-1)$为多项式．

【定理 1】任何有理式都可利用辗转相除法化成连分式．

【证明】设分子是 $P_0(x)$，分母是 $Q_0(x)$，若 $P_0(x)$ 的阶高于 $Q_0(x)$ 的阶，当 $P_0(x)$ 能整除 $Q_0(x)$ 时，显然满足定理要求；若不能整除 $Q_0(x)$，设商为 $b_0(x)$，余数为 $P_1(x)$，显然 $P_1(x)$ 的阶低于 $Q(x)$ 的阶，即

$$\frac{P(x)}{Q(x)} = b_0(x) + \frac{P_1(x)}{Q_0(x)}\,.$$

由于

$$b_0(x) + \frac{P_1(x)}{Q_0(x)} = b_0(x) + \frac{1}{\dfrac{Q_0(x)}{P_1(x)}}$$

$$= b_0(x) + \cfrac{1}{b_1(x) + \cfrac{Q_1(x)}{P_1(x)}}$$

$$= b_0(x) + \cfrac{1}{b_1(x) + \cfrac{1}{\cfrac{P_1(x)}{Q_1(x)}}}$$

因此，上面的过程可一直继续下去，直到余数为 0 或者为常数为止.

当 $P_0(x)$ 的阶低于 $Q_0(x)$ 的阶时，$\cfrac{P_0(x)}{Q_0(x)} = \cfrac{1}{\cfrac{Q_0(x)}{P_0(x)}}$，此时 $b_0 = 0$，同样可利用上述方法，使得余数为零或余数为常数，满足定理要求. 证毕.

上面所有的算法称为辗转相除法.

【例 1】 将有理分式

$$R(x) = \frac{2x^4 + 45x^3 + 381x^2 + 1353x + 1511}{x^3 + 21x^2 + 157x + 409}$$

化成连分式.

【解】 计算过程如下.

（1）求 $2x^4 + 45x^3 + 381x^2 + 1353x + 1511$ 除以 $x^3 + 21x^2 + 157x + 409$ 的商和余数.

$$\begin{array}{r} \qquad\qquad\qquad\qquad\qquad\qquad 2x+3 \\ x^3+21x^2+157x+409 \overline{\left) \begin{array}{r} 2x^4+45x^3+381x^2+1353x+1511 \\ \underline{2x^4+42x^3+314x^2+\ 818x\ \ \ \ \ \ } \\ 3x^3+\ 67x^2+\ 535x+1511 \\ \underline{3x^3+\ 63x^2+\ 471x+1227} \\ 4x^2+\ \ 64x+\ 284 \end{array} \right.} \end{array}$$

即

$$R(x) = 2x + 3 + \frac{4x^2 + 64x + 284}{x^3 + 21x^2 + 157x + 409}$$

$$= 2x + 3 + \cfrac{1}{\cfrac{x^3 + 21x^2 + 157x + 409}{4x^2 + 64x + 284}}$$

（2）求 $x^3 + 21x^2 + 157x + 409$ 除以 $4x^2 + 64x + 284$ 的商和余数.

$$
\begin{array}{r}
\dfrac{1}{4}x+\dfrac{5}{4} \\
4x^2+64x+284\overline{\left)\begin{array}{r} x^3+21x^2+157x+409 \\ \underline{x^3+16x^2+\ \ 71x\ \ \ \ \ \ \ \ } \\ 5x^2+\ \ 86x+409 \\ \underline{5x^2+\ \ 80x+355} \\ 6x+\ \ 54 \end{array}\right.}
\end{array}
$$

故

$$
\begin{aligned}
R(x) &= 2x+3+\cfrac{1}{\dfrac{1}{4}x+\dfrac{5}{4}+\dfrac{64+54}{4x^2+64x+284}} \\
&= 2x+3+\cfrac{1}{\dfrac{1}{4}x+\dfrac{5}{4}+\cfrac{1}{\dfrac{4x^2+64x+284}{6x+54}}}
\end{aligned}
$$

（3）求 $4x^2+64x+284$ 除以 $6x+54$ 的商和余数.

$$
\begin{array}{r}
\dfrac{2}{3}x+\dfrac{14}{3} \\
6x+54\overline{\left)\begin{array}{r} 4x^2+64x+284 \\ \underline{4x^2+36x\ \ \ \ \ \ } \\ 28x+284 \\ \underline{28x+252} \\ 32 \end{array}\right.}
\end{array}
$$

$$
\begin{aligned}
R(x) &= 2x+3+\cfrac{1}{\dfrac{1}{4}x+\dfrac{5}{4}+\cfrac{1}{\dfrac{2}{3}x+\dfrac{14}{3}+\dfrac{32}{6x+54}}} \\
&= 2x+3+\cfrac{1}{\dfrac{1}{4}x+\dfrac{5}{4}+\cfrac{1}{\dfrac{2}{3}x+\dfrac{14}{3}+\cfrac{1}{\dfrac{6x+54}{32}}}}
\end{aligned}
$$

（4）计算 $6x+54$ 除以 32 的商和余数.

$$
\begin{array}{r}
\dfrac{3}{16}x \\
32\overline{\left)\begin{array}{r} 6x+54 \\ \underline{6x\ \ \ \ } \\ 54 \end{array}\right.}
\end{array}
$$

此时余数已为常数，计算完毕.

由上述计算得

$$R(x)=2x+3+\cfrac{1}{\cfrac{1}{4}x+\cfrac{5}{4}+\cfrac{1}{\cfrac{2}{3}x+\cfrac{14}{3}+\cfrac{1}{\cfrac{3}{16}x+\cfrac{3}{2}}}}$$

$$=2x+3+\cfrac{4}{x+5+\cfrac{1}{\cfrac{2}{3}x+\cfrac{14}{3}+\cfrac{1}{\cfrac{3}{16}x+\cfrac{3}{2}}}}$$

$$=2x+3+\cfrac{4}{x+5+\cfrac{4}{\cfrac{2}{3}x+\cfrac{14}{3}+\cfrac{1}{\cfrac{3}{16}x+\cfrac{3}{2}}}}$$

$$=2x+3+\cfrac{4}{x+5+\cfrac{6}{x+7+\cfrac{3/2}{\cfrac{3}{16}x+\cfrac{3}{2}}}}$$

$$=2x+3+\cfrac{4}{x+5+\cfrac{6}{x+7+\cfrac{24}{3x+9}}}$$

注意：最后四个等式可不必考虑，即计算到第一个等式就可结束计算.

8.1.2 连分式计算

如果已知 x 值，如何计算连分式？下面的递推公式给出了连分式算法.

对于式（8.1-1），令 $g_n=b_n$，则

$$g_{n-1}=b_{n-1}+\frac{a_n}{b_n}=b_{n-1}+\frac{a_n}{g_n}$$

$$g_{n-2}=b_{n-2}+\frac{a_{n-1}}{g_{n-1}}$$

$$\cdots$$

归纳得

$$\begin{cases} g_n = b_n \\ g_k = b_k + \dfrac{a_{k+1}}{g_{k+1}} \qquad k = n-1, n-2, \cdots, 0 \\ g_0 = R_n \end{cases} \tag{8.1-2}$$

除了用式（8.1-2）计算连分式以外，还可用另一递推算法来计算连分式.

【定理 2】对于连分式（8.1-1），若用递推公式

$$\begin{cases} p_k = b_k p_{k-1} + a_k p_{k-2} \\ q_k = b_k q_{k-1} + a_k q_{k-2} \qquad k = 1, 2, \cdots, n \\ p_{-1} = 1,\ q_{-1} = 0,\ p_0 = b_0,\ q_0 = 1 \end{cases} \tag{8.1-3}$$

则

$$R_n = \frac{p_n}{q_n}.$$

【证明】现用数学归纳法证明.

当 $n = 1$ 时，

$$R_1 = \frac{p_1}{q_1} = b_0 + \frac{a_1}{b_1} = \frac{b_0 b_1 + a_1}{b_1},$$

由式（8.1-3）有

$$p_1 = b_1 p_0 + a_1 p_{-1} = b_0 b_1 + a_1$$
$$q_1 = b_1 q_0 + a_1 q_{-1} = b_1$$
$$R_1 = \frac{b_0 b_1 + a_1}{b_1} = \frac{p_1}{q_1}$$

定理成立.

现令 $n = m$ 时定理成立，即

$$R_m = \frac{b_m p_{m-1} + a_m p_{m-2}}{b_m q_{m-1} + a_m q_{m-2}}.$$

另一方面，由 R_{m+1} 的定义可知，只须将 $b_m + \dfrac{a_{m+1}}{b_{m+1}}$ 代替 b_m，就可给出 R_{m+1}，即

$$R_{m+1} = \frac{\left(b_m + \dfrac{a_{m+1}}{b_{m+1}}\right) p_{m-1} + a_m p_{m-2}}{\left(b_m + \dfrac{a_{m+1}}{b_{m+1}}\right) q_{m-1} + a_m q_{m-2}}$$

$$= \frac{b_m p_{m-1} + \dfrac{a_{m+1}}{b_{m+1}} p_{m-1} + a_m p_{m-1}}{b_m q_{m-1} + \dfrac{a_{m+1}}{b_{m+1}} q_{m-1} + a_m q_{m-2}}$$

$$
\begin{aligned}
&= \frac{b_{m+1}(b_m p_{m-1} + a_m p_{m-2}) + a_{m+1} p_{m-1}}{b_{m+1}(b_m q_{m-1} + a_m q_{m-2}) + a_m q_{m-1}} \\
&= \frac{b_{m+1} p_m + a_{m+1} p_{m-1}}{b_{m+1} q_m + a_{m+1} q_{m-1}} \\
&= \frac{p_{m+1}}{q_{m+1}}
\end{aligned}
$$

综上可知定理成立.

8.2 有理插值

有理式或有理函数就是分子、分母都是多项式的算式，用有理式逼近函数且在节点处的值与被逼近函数的值相等时，这种有理函数称为有理插值函数.

8.2.1 有理插值函数

【定义 1】 若序列 $\{v_k(x)\}$ 满足

$$
\begin{cases} v_k(x) = \dfrac{x - x_{k-1}}{v_{k-1}(x) - v_{k-1}(x_{k-1})} & k = 1, 2, \cdots \\ v_0(x) = f(x) \end{cases} \tag{8.2-1}
$$

则称 $v_k(x)$ 为函数 $f(x)$ 的 k 阶反差商.

由式（8.2-1）可得

$$
v_{k-1}(x) = v_{k-1}(x_{k-1}) + \frac{x - x_{k-1}}{v_k(x)}, \quad k = 1, 2, \cdots
$$

又因为 $v_0(x) = f(x)$，于是有

$$
\begin{aligned}
f(x) &= v_0(x) \\
&= v_0(x_0) + \frac{x - x_0}{v_1(x)} \\
&= v_0(x_0) + \cfrac{x - x_0}{v_1(x_1) + \cfrac{x - x_1}{v_2(x)}} \\
&\cdots \\
&= v_0(x_0) + \cfrac{x - x_0}{v_1(x_1) + \cfrac{x - x_1}{v_2(x_2) + \ddots + \cfrac{x - x_{n-1}}{v_n(x_n) + \cfrac{x - x_n}{v_{n+1}(x)}}}}
\end{aligned}
$$

去掉 $\dfrac{x-x_n}{v_{n+1}(x)}$，并记

$$R_n(x)=v_0(x_0)+\cfrac{x-x_0}{v_1(x_1)+\cfrac{x-x_1}{v_2(x_2)+\ddots+\cfrac{x-x_{n-1}}{v_n(x_n)}}} \tag{8.2-2}$$

【定理 1】当 x_k 互异时，$R_n(x_k)=f(x_k)\,(k=0,1,\cdots,n)$，即 $R_n(x)$ 是 n 阶有理插值函数.

【证明】按定义有

$$f(x_k)=v_0(x_0)+\cfrac{x_k-x_0}{v_1(x_1)+\cfrac{x_k-x_1}{v_2(x_2)+\ddots+v_{k-1}(x_{k-1})+\cfrac{x_k-x_{k-1}}{v_k(x_k)}}}$$

当 $k\leqslant n$ 时，

$$\begin{aligned}R_n(x_k)&=v_0(x_0)+\cfrac{x_k-x_0}{v_1(x_1)+\cfrac{x_k-x_1}{v_2(x_2)+\ddots+v_{k-1}(x_{k-1})+\cfrac{x_k-x_{k-1}}{v_k(x_k)}}}\\&=f(x_k)\end{aligned}$$

8.2.2　反差商递推计算公式

为了计算反差商，可制作如表 8.1 所示的反差商表.

表 8.1　反差商表

x	$f(x)=v_0(x)$	$v_1(x_1)$	…	$v_i(x_i)$	…	$v_{n-1}(x_{n-1})$	$v_n(x_n)$
x_0	$v_0(x_0)$						
x_1	$v_0(x_1)$	$v_1(x_1)$					
⋮	⋮	⋮	⋱				
x_i	$v_0(x_i)$	$v_1(x_i)$	…	$v_i(x_i)$			
⋮	⋮	⋮	…	⋮	⋱		
x_{n-1}	$v_0(x_{n-1})$	$v_1(x_{n-1})$	…	$v_i(x_{n-1})$	…	$v_{n-1}(x_{n-1})$	
x_n	$v_0(x_n)$	$v_1(x_n)$	…	$v_i(x_n)$	…	$v_{n-1}(x_n)$	$v_n(x_n)$

按 k 阶反差商的定义有

$$v_k(x_j)=\frac{x_j-x_{k-1}}{v_{k-1}(x_j)-v_{k-1}(x_{k-1})},\quad k=1,2,\cdots,n,\quad j=k,k+1,\cdots,n \tag{8.2-3}$$

反差商是一个二维表，但式（8.2-3）看不出计算结果的行列位置，且不便于存放．下面是能看出其行列位置并便于程序设计的递推公式：

$$v_{j,k}(x_j)=\frac{x_j-x_{k-1}}{v_{j,k-1}-v_{k-1,k-1}},\quad k=1,2,\cdots,n,\quad j=k,k+1,\cdots,n \tag{8.2-4}$$

8.2.3 有理插值计算过程及计算实例

1．计算过程

（1）给出节点值及节点函数值，利用式（8.2-3）计算 $v_{k,k}$，$v_{y0}=f(x_j)$ 已知；

（2）根据 $R_n(x)$ 的定义，用以下递推公式及式（8.2-2）计算 $R_n(x)$ 的值：

$$\begin{cases} g_n=v_{n,n} \\ g_k=v_{k,k}+\dfrac{x-x_k}{g_{k+1}} \qquad k=n-1,n-2,\cdots,0 \end{cases} \tag{8.2-5}$$

2．计算实例

【例 1】作一有理插值函数，使

$$R_4(0)=1,\quad R_4(1)=\frac{1}{2},\quad R_4(2)=\frac{1}{5},\quad R_4(3)=\frac{1}{10},\quad R_4(4)=\frac{1}{17},$$

并计算 $R_4(2.5)$ 的值．

【解】按式（8.2-4）计算得到结果如表 8.2 所示．

表 8.2 反差商计算结果

x	$v_{j,0}$	$v_{j,1}$	$v_{j,2}$	$v_{j,3}$	$v_{j,4}$
0	1				
1	$\frac{1}{2}$	-2			
2	$\frac{1}{5}$	$-\frac{5}{2}$	-2		
3	$\frac{1}{10}$	$-\frac{10}{3}$	$-\frac{3}{2}$	2	
4	$\frac{1}{17}$	$-\frac{17}{4}$	$-\frac{4}{3}$	3	1

有理插值函数为

$$R_4(x)=1+\cfrac{x-0}{-2+\cfrac{x-1}{-2+\cfrac{x-2}{2+\cfrac{x-3}{1}}}},$$

经整理得

$$R_4(x)=\frac{1}{1+x^2}.$$

一般情况下不需要整理.

当 $x=2.5$ 时，按上面的计算结果有

$$g_4=1,$$

$$g_3=2+\frac{2.5-3}{1}=1.5,$$

$$g_2=-2+\frac{2.5-2}{1.5}=-\frac{5}{3},$$

$$g_1=-2+\frac{2.5-1}{-\frac{5}{3}}=-\frac{29}{10},$$

$$g_0=1+\frac{2.5-0}{-\frac{29}{10}}=\frac{4}{29}\doteq 0.137931.$$

本例不将 x 直接代入 $R_4(x)=\dfrac{1}{1+x_2}$，而代入连分式，使计算更具有一般性.

8.2.4　有理插值的逐次计算法

有理插值函数既然可写成连分式，当然也可用逐次计算法计算其值．逐次插值的特点是可直接利用前一次的计算结果.

仿照连分式逐次计算公式可直接得到逐次有理插值计算公式：

$$\begin{cases} p_k(x)=v_{k,k}p_{k-1}+(x-x_{k-1})p_{k-2} \\ q_k(x)=v_{k,k}q_{k-1}+(x-x_{k-1})q_{k-2} \qquad k=1,2,\cdots,n \\ p_{-1}=1,\ p_0=v_{0,0} \\ q_{-1}=0,\ q_0=1 \end{cases} \tag{8.2-6}$$

$$R_k=\frac{p_k}{q_k}$$

8.2.5　有理插值逐次计算法的计算过程和计算实例

1．计算过程

（1）利用反差商计算公式计算 $v_{i,i}$（$i=1,2,\cdots,n$）；

（2）利用式（8.2-6）计算 $p_k(x)$，$q_k(x)$（$k=1,2,\cdots,n$）；

（3）计算 $R_k=\dfrac{p_k}{q_k}$.

2．计算实例

【例 1】$\csc 1^\circ$，$\csc 2^\circ$，$\csc 3^\circ$，$\csc 4^\circ$ 的值如下：

x	1°	2°	3°	4°
$\csc x$	57.298667	28.653706	19.107321	14.335588

作一有理插值函数，用连分式和逐次插值计算 $\csc 1.5^\circ$ 的近似值.

【解】按反差商计算公式（8.2-4）计算出的 $v_{j,k}$ 值如表 8.3 所示.

表 8.3 反差商的计算结果

x	$v_{j,0}$	$v_{j,1}$	$v_{j,2}$	$v_{j,3}$
1	57.298691			
2	28.653706	−0.034910		
3	19.107323	−0.052368	−57.281223	
4	14.335588	−0.069827	−57.278312	344.020996

直接利用连分式计算得

$$
\begin{aligned}
R_3(1.5) &= 57.298691 + \cfrac{1.5-1}{-0.034910 + \cfrac{1.5-2}{-57.281223 + \cfrac{1.5-3}{344.020996}}} \\
&= 57.298691 + \cfrac{0.5}{-0.034910 + \cfrac{-0.5}{-57.281223 + \cfrac{-1.5}{344.020996}}} \\
&= 57.298691 + \cfrac{0.5}{-0.034910 + \cfrac{-0.5}{-57.281223 - 0.002907}} \\
&= 57.298691 + \cfrac{0.5}{-0.034910 + \cfrac{-0.5}{-57.285583}} \\
&= 57.298691 + \cfrac{0.5}{-0.034910 + 0.008728} \\
&= 57.298691 + \cfrac{0.5}{-0.026182} \\
&= 57.298691 - 19.097090 \\
&= 38.201601
\end{aligned}
$$

用逐次有理插值计算，步骤及结果如下.

（1）利用反差商计算公式生成反差商表，计算结果同表 8.3；

（2）按逐次有理插值计算公式计算 p_1，p_2，p_3，q_1，q_2，q_3．为了表现逐次插值计算，特给出 R_1，R_2，R_3.

$$v_{0,0}=p_0=57.298691$$
$$q_0=1$$
$$p_1=v_{1,1}p_0+(x-x_0)p_{-1}=-0.034910\times 57.298691+0.5\times 1=-1.500305$$
$$q_1=v_{1,1}q_0+(x-x_0)q_{-1}=-0.034910+0.5\times 0=-0.034910$$
$$R_1=\frac{p_1}{q_1}=42.976200$$
$$\begin{aligned}p_2&=v_{2,2}p_1+(x-x_1)p_0\\&=-57.281223\times(-1.500305)-0.5\times 57.298691=57.289936\end{aligned}$$
$$\begin{aligned}q_2&=v_{2,2}q_1+(x-x_1)q_0\\&=-57.281223\times(-0.034910)-0.5\times 1=1.499695\end{aligned}$$
$$R_2=\frac{p_2}{q_2}=38.201061$$
$$\begin{aligned}p_3&=v_{3,3}p_2+(x-x_2)p_1\\&=344.020996\times 57.289936-1.5\times 1.500305=19711.191406\end{aligned}$$
$$\begin{aligned}q_3&=v_{3,3}q_2+(x-x_2)q_1\\&=344.020996\times 1.499695+0.5\times 0.034910=515.97882\end{aligned}$$
$$R_3=\frac{p_3}{q_3}=38.201551$$

直接调用 csc 函数计算得到 $\csc 1.5^\circ=38.201550$．用三阶代数插值所得结果为 39.687381．由此可见有理插值函数比代数多项式逼近 $\csc x$ 函数好，原因是 $\csc x$ 函数在 $x=0$ 时是无界的．

两种算法略有差异，原因是计算误差引起的．

8.2.6　误差估计

用连分式对有理函数做插值计算的计算量约为逐次插值计算的四分之一，但连分式计算难以估计误差．

【定理 2】若节点 x_i（$i=0,1,\cdots,n$）互异，$f(x_i)$ 已知，设 $f(x)$ 在 $x\in[a,b]$ 上有 $n+1$ 阶连续导数，且 $q_n(x)\neq 0$，则

$$f(x)-\frac{p_n(x)}{q_n(x)}=\frac{(x-x_0)\cdots(x-x_n)}{(n+1)!q_n^2(x)}(f(\xi)q_n^2(x))^{(n+1)}.$$

上式中 $\dfrac{p_n(x)}{q_n(x)}$ 是过节点 x_i 的有理插值函数．

【证明】当 $x=x_i$ 时，等式左右端均为 0，定理成立．

当 $x\neq x_i$ 时，作辅助函数

$$\varphi(t)=f(t)q_n^2(t)-p_n(t)q_n(t)-\frac{f(x)q_n^2(x)-p_n(x)q_n(x)}{(x-x_0)\cdots(x-x_n)}(t-t_0)\cdots(t-t_n)$$

显然有

$\varphi(x_i)=0$， $i=0,1,\cdots,n$

$\varphi(x)=0$

即 $\varphi(t)$ 有 $n+2$ 个零点，反复利用罗尔定理可知，存在 ξ 使得

$$\varphi^{(n+1)}(\xi)=0\ .$$

由于 $p_n(t)q_n(t)$ 是不超过 n 次的多项式，所以有

$$0=\varphi^{(n+1)}(\xi)\doteq(f(\xi)q_n^2(\xi))^{(n+1)}-\frac{f(x)q_n^2(x)-p_n(x)q_n(x)}{(x-x_0)\cdots(x-x_n)}(n+1)!$$

整理后可得要证明的算式.

习题 8

1．将有理函数 $\dfrac{x+1}{6x^2-5x+1}$ 化成连分式形式.

2．将有理函数 $\dfrac{5x^3-3x^2-14x}{x^2+x+2}$ 化成连分式形式.

3．给定下列一组数值，作有理插值多项式.

x	0	1	2	3
y	4	2	3	7

4．取 $x_0=0$， $x_1=0.25$， $x_2=0.5$， $x_3=0.75$， $x_4=1$，对函数 $y=\ln(1+x)$, $x\in[0,1]$ 作有理插值逼近.

5．取 $x_0=0$， $x_1=0.25$， $x_2=0.5$， $x_3=0.75$， $x_4=1$，用逐次有理插值逼近公式逼近 $y=\mathrm{e}^x$.

第 9 章　数值微积分

在高等数学中介绍微分的篇幅不是很多，原因是对于所有基本初等函数及各种各样的复合函数，都有对应的可操作性极强的算法，只要按部就班就能得到理论解．不过要得到具有指定精度的导数的数值解，哪怕是一阶导数都很难，要编写具有足够精度的通用程序更难．

在高等数学中，介绍积分的算法很多，分类也很细，不过已知被积函数，要找到对应的行之有效的算法却很难，更麻烦的是绝大多数被积函数算不出定积分的理论值，即找不到原函数．

实际上，工程物理问题所涉及的被积函数常常具有以下特点．

（1）只能用图表表示，函数值通常是离散的具有测试误差的测量值；

（2）难以用 Newton-Leibnitz 公式得到理论解；

（3）只需要具有一定精度的一阶导数和二阶导数的数值解．也就是说，工程物理问题中，通常不是用高等数学教材中的方法求微积分的理论解，而是用计算机计算数值解．

9.1　数值积分基本方法

目前，数值积分计算公式虽多，但基本思想相同．

9.1.1　一般数值积分公式

所谓 $\int_a^b f(x)\mathrm{d}x$ 的数值积分的数值解法，是指用被积函数 $f(x)$ 的一组节点 $a \leqslant x_0 < \cdots < x_n \leqslant b$ 的值 $f(x_i)\ (i=0,1,\cdots,n)$ 的某种线性组合，即用

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x &\approx A_0 f(x_0) + A_1 f(x_1) + \cdots + A_n f(x_n) \\ &= \sum_{i=0}^{n} A_i f(x_i)\end{aligned} \tag{9.1-1}$$

表示的算式表示积分，式（9.1-1）和理论解的关系为

$$\int_a^b f(x)\mathrm{d}x = \sum_{i=0}^{n} A_i f(x_i) + R(f) \tag{9.1-2}$$

式（9.1-1）中 A_i 称为求积系数，式（9.1-2）中 $R(f)$ 称为求积余项，也就是求积公式的截断误差．当 $R(f)=0$ 时，$\sum_{i=0}^{n} A_i f(x_i)$ 为理论解．

9.1.2 构造求积公式的基本方法

一般情况下，被积函数 $f(x)$ 的原函数 $F(x)$ 理论上是存在的，但却找不到，为此常用较简单的性质良好的可以找到原函数的函数 $g(x)$ 代替 $f(x)$ 进行积分．这些替代函数可以是代数插值多项式，可以是样条函数，也可以是有理插值函数，还可以是其他多项式或三角函数．显然，代数插值多项式最简单，性质也最为良好．

由于 n 阶 Lagrange 插值多项式

$$L_n(x)=\sum_{i=0}^{n}\left(\sum_{\substack{j=0\\ j\neq i}}^{n}\frac{x-x_j}{x_i-x_j}\right)y_i$$

经积分后的积分结果和式（9.1-1）一致，故常选 $L_n(x)$ 代替 $f(x)$．当 $L_n(x)$ 是等距节点的插值多项式时，积分公式称为 Newton 积分公式；当节点选取某一高一阶的（加权）正交多项式的零点时，积分公式称为 Gauss 积分公式；若被积函数用 $L_1(x)$ 代替，并使用外推法，则积分公式称为 Romberg 积分公式．

本章只介绍 Newton 积分、Gauss 积分和 Romberg 积分，不介绍利用样条函数、有理插值函数及其他函数的值的积分．

当 $f(x)$ 用 $L_n(x)=\sum_{i=0}^{n}l_i(x)f(x_i)$ 代替时，数值积分公式为

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x&=\int_a^b L_n(x)\mathrm{d}x+R(f)\\&=\int_a^b\sum_{i=0}^{n}l_i(x)f(x_i)\mathrm{d}x+R(f)\\&=\sum_{i=0}^{n}\int_a^b l_i(x)f(x_i)\mathrm{d}x+R(f)\\&=\sum_{i=0}^{n}A_i f(x_i)+R(f)\end{aligned}$$

9.1.3 代数精确度

对于用多项式代替被积函数 $f(x)$ 的积分，显然当 $f(x)$ 是一般函数时难以获得理论值，若 $f(x)$ 是多项式时能否得到理论值呢？

【定义】如果数值求积公式 $\sum_{i=0}^{n}A_i f(x_i)$ 对于任意不高于 m 次的多项式能得到理论解，当 $f(x)$ 的阶为 $m+1$ 时只能得到近似解，则称数值求积公式的代数精确度为 m．

代数精确度是数值积分公式衡量积分公式积分精确度的标准．显然，代数精确度愈高，截断误差愈小．

9.2　数值积分基本方法

所有用等距 Lagrange 插值多项式代替被积函数 $f(x)$ 的积分都称为 Newton 积分，当 Lagrange 插值多项式的阶为 1 和 2 时，称为低阶 Newton 积分.

9.2.1　梯形积分公式

梯形积分公式就是用一阶 Lagrange 插值多项式代替被积函数的积分公式. 梯形积分公式的得名，来自算式的几何意义.

当 Lagrange 插值多项式的阶为 1，且 $x_0=a$，$x_1=b$ 时，积分公式为

$$\begin{aligned} S_1 &= \int_a^b L_1(x)\mathrm{d}x \\ &= \int_a^b \left(\frac{x-b}{a-b} f(a) + \frac{x-a}{b-a} f(b) \right) \mathrm{d}x \\ &= \frac{1}{2}\big(f(a)+f(b)\big)(b-a) \end{aligned} \qquad (9.2\text{-}1)$$

当 $f(x)\geqslant 0$ 时，式（9.2-1）的几何图形如图 9.1 所示.

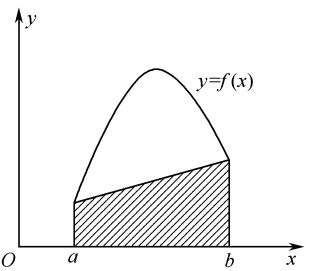

图 9.1　梯形积分的几何图形

图 9.1 中的阴影部分为一梯形，其面积等于 $\frac{1}{2}(f(a)+f(b))(b-a)$，这正是梯形积分公式名字的由来.

9.2.2　梯形积分公式的截断误差

所谓截断误差，是指 $\int_a^b f(x)\mathrm{d}x - \int_a^b L_n(x)\mathrm{d}x$ 之差. 由图 9.1 可以看出，一般情况下梯形公式的截断误差不小.

【定理 1】 若 $f(x)$，$x\in[a,b]$ 至少存在二阶导数，则梯形公式的截断误差为

$$R(f) = -\frac{f''(\xi)}{12}(b-a)^3 .$$

【证明】

$$\begin{aligned} \int_a^b f(x)\mathrm{d}x - \int_a^b L_1(x)\mathrm{d}x &= \int_a^b \big(f(x)-L_1(x)\big)\mathrm{d}x \\ &= \int_a^b \frac{f''(\eta)}{2}(x-a)(x-b)\mathrm{d}x \end{aligned}$$

因为 $(x-a)(x-b)\leqslant 0$，$x\in[a,b]$，由积分中值定理可知，存在 $\xi\in(a,b)$，使得

$$\begin{aligned} \int_a^b \frac{f''(\eta)}{2}(x-a)(x-b)\mathrm{d}x &= \frac{f''(\xi)}{2}\int_a^b (x-a)(x-b)\mathrm{d}x \\ &= -\frac{f''(\xi)}{12}(b-a)^3 \end{aligned} \qquad (9.2\text{-}2)$$

9.2.3 复合梯形积分公式

从梯形积分公式的截断误差公式可以看出，当积分区间 $(b-a)$ 较大时，或二阶导数中最大值较大时，截断误差会相当大．但梯形积分公式的计算公式简单，计算方便．为了降低截断误差，人们常使用复合（也称为复化）积分公式．所谓复合梯形积分公式，就是将积分区间分成 n 等份，各子区间都用梯形积分公式计算积分，然后用各子区间积分和表示总的积分，低阶 Newton 积分及低阶 Gauss 积分都常用复合积分公式．

对于 $\int_a^b f(x)\mathrm{d}x$，复合梯形积分公式的推导过程及公式如下．

设 $x_i = a + ih\ (i = 0,1,\cdots,n)$，$h = \dfrac{b-a}{n}$（通常称 h 为步长），则

$$\begin{aligned}\int_{x_{i-1}}^{x_i} L_1(x)\mathrm{d}x &= \int_{a+(i-1)h}^{a+ih} L_1(x)\mathrm{d}x \\ &= \frac{1}{2}\big(f(x_{i-1}) + f(x_i)\big)h\end{aligned}$$

$$\begin{aligned}\sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} L_1(x)\mathrm{d}x &= \frac{1}{2}\sum_{i=1}^{n}\big(f(x_{i-1}) + f(x_i)\big)h \\ &= \left(\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{n-1} f(x_i)\right)h \\ &= \left(\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{n-1} f(a+ih)\right)h\end{aligned} \tag{9.2-3}$$

式（9.2-3）就是复合梯形积分公式．

9.2.4 复合梯形积分公式截断误差

由梯形积分公式的截断误差可直接推导出复合梯形积分公式的截断误差．

$$\begin{aligned}R(f) &= \int_a^b f(x)\mathrm{d}x - \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} L_1(x)\mathrm{d}x \\ &= \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} f(x)\mathrm{d}x - \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} L_1(x)\mathrm{d}x \\ &= \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} (f(x) - L_1(x))\mathrm{d}x \\ &= \sum_{i=1}^{n} -\frac{f''(\xi_i)}{12}h^3, \qquad \xi_i \in (x_{i-1}, x_i) \\ &= -n\frac{f''(\xi)}{12}h^3, \qquad f''(\xi) = \frac{\sum_{i=1}^{n} f''(\xi_i)}{n} \\ &= -\frac{f''(\xi)}{12}h^2(b-a)\end{aligned} \tag{9.2-4}$$

从式（9.2-4）可以看出，当 $h \to 0$ 时，$R(f) = 0$．可见复合梯形公式是收敛的．式（9.2-4）所示公式简单，算式又收敛，因而是一实用公式．

9.2.5　复合梯形积分公式计算过程和计算实例

1．计算过程

（1）计算步长 $h = \dfrac{b-a}{n}$，n 为分段数；

（2）计算 $S = \dfrac{1}{2}f(a) + \dfrac{1}{2}f(b) + \sum_{i=1}^{n} f(x_i) = \dfrac{1}{2}f(a) + \dfrac{1}{2}f(b) + \sum_{i=1}^{n} f(a+ih)$；

（3）计算 $S_t = S \times h$．

2．计算实例

【例 1】计算当 n 分别为 1, 2, 4 时的积分：$I = \int_0^1 \dfrac{\mathrm{d}x}{1+x^2}$．

【解】计算公式为

$$S_t = h\left(\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{n-1} f(x_i)\right) = \frac{1}{n}\left(\frac{1}{2} + \frac{1}{2}\times\frac{1}{2} + \sum_{i=1}^{n-1}\frac{1}{1+(i/n)^2}\right).$$

计算结果见表 9.1、表 9.2 和表 9.3．

表 9.1　$n=1$ 时积分计算结果

n	h	x_0	x_1	f_0	f_1	S_t
1	1	0	1	1	0.5	0.75

表 9.2　$n=2$ 时积分计算结果

n	h	x_0	x_1	x_2	f_0	f_1	f_2	S_t
2	0.5	0	0.5	1	1	0.8	0.5	0.775

表 9.3　$n=4$ 时积分计算结果

n	h	f_0	f_1	f_2	f_3	f_4	S_t
4	0.25	0	0.94118	0.8	0.64	0.5	0.78279

9.2.6　Simpson 积分公式

梯形积分公式用直线代替曲线 $f(x)$ 计算定积分，虽然计算公式简单，稳定性也好，但是计算精度较低．如果用二次曲线代替曲线 $f(x)$，计算精度将大大提高，用二次曲线代替曲线，即用二阶等距 Lagrange 插值多项式代替 $f(x)$ 的定积分，称为 Simpson 积分．当 $y = L_2(x)$ 时，$L_2(x)$ 若是二次曲线，则只能是抛物线，所以 Simpson 积分也称为抛物线积分．

下面是 Simpson 积分公式的推导过程.

$$S=\int_a^b L_2(x)\mathrm{d}x=A_0 f(x_0)+A_1 f(x_1)+A_2 f(x_2)\,,\quad x_0=a\,,\quad x_1=\frac{a+b}{2}\,,\quad x_2=b$$

$$A_0=\int_a^b \frac{\left(x-\dfrac{a+b}{2}\right)(x-b)}{\left(a-\dfrac{a+b}{2}\right)(a-b)}\mathrm{d}x=\frac{1}{6}(b-a)$$

$$A_1=\int_a^b \frac{(x-a)(x-b)}{\left(\dfrac{a+b}{2}-a\right)\left(\dfrac{a+b}{2}-b\right)}\mathrm{d}x=\frac{2}{3}(b-a)$$

$$A_2=\int_a^b \frac{(x-a)\left(x-\dfrac{a+b}{2}\right)}{(b-a)\left(b-\dfrac{a+b}{2}\right)}\mathrm{d}x=\frac{1}{6}(b-a)$$

由此可知 Simpson 计算公式为

$$S=\frac{1}{6}(b-a)f(a)+\frac{2}{3}(b-a)f\left(\frac{a+b}{2}\right)+\frac{1}{6}(b-a)f(b) \tag{9.2-5}$$

9.2.7 Simpson 积分的代数精确度

由于 $L_2(x)$ 是二次代数插值多项式，当 $f(x)$ 是二次或二次以下多项式时，$L_2(x)=f(x)$，自然，此时 $\int_a^b L_2(x)\mathrm{d}x=\int_a^b f(x)\mathrm{d}x$．现假设 $f(x)$ 是任意三次多项式时，由定积分性质可知，只要

$$\int_a^b L_2(x)\mathrm{d}x=\int_a^b x^3\mathrm{d}x\,,$$

则 $f(x)$ 是任意三次多项式时上式仍然成立．当 $f(x)=x^3$ 时，

$$\begin{aligned}\int_a^b L_2(x)\mathrm{d}x&=\frac{1}{6}(b-a)f(a)+\frac{2}{3}(b-a)f\left(\frac{a+b}{2}\right)+\frac{1}{6}(b-a)f(b)\\&=\frac{1}{6}(b-a)a^3+\frac{2}{3}(b-a)\left(\frac{a+b}{2}\right)^3+\frac{1}{6}(b-a)b^3\\&=\frac{1}{4}(b^4-a^4)\\&=\int_a^b x^3\mathrm{d}x\end{aligned}\quad .$$

当 $f(x)$ 是任意四阶多项式时，$\int_a^b x^4\mathrm{d}x=\frac{1}{5}(b^5-a^5)\neq\int_a^b L_2(x)\mathrm{d}x$，所以 Simpson 积分的代

数精确度为 3.

9.2.8　Simpson 积分公式的截断误差

很容易证明梯形积分公式的代数精确度为 1，前面已证明 Simpson 积分公式的代数精确度为 3，因此 Simpson 积分的截断误差应比梯形积分的小，即收敛性质比前者好.

【定理 2】若 $y=f(x)$ 在 $x\in[a,b]$ 上至少有四阶导数，则

$$\int_a^b f(x)\mathrm{d}x-\int_a^b L_2(x)\mathrm{d}x=-\frac{f^{(4)}(\xi)}{2880}(b-a)^5 \tag{9.2-6}$$

【证明】作带重节点的三阶 Newton 插值多项式 $N_3(x)$，使

$$N_3(a)=f(a),\quad N_3\left(\frac{a+b}{2}\right)=f\left(\frac{a+b}{2}\right),\quad N_3'\left(\frac{a+b}{2}\right)=f'\left(\frac{a+b}{2}\right),\quad N_3(b)=f(b).$$

显然，$L_2(x)$ 同时是 $f(x)$ 和 $N_3(x)$ 的二阶代数插值多项式，所以

$$\int_a^b L_2(x)\mathrm{d}x=\int_a^b N_3(x)\mathrm{d}x$$

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x-\int_a^b L_2(x)\mathrm{d}x&=\int_a^b f(x)\mathrm{d}x-\int_a^b N_3(x)\mathrm{d}x\\&=\int_a^b\big(f(x)-N_3(x)\big)\mathrm{d}x\\&=\int_a^b\frac{f^{(4)}(\eta)}{4!}(x-a)\left(x-\frac{a+b}{2}\right)^2(x-b)\mathrm{d}x\\&=\frac{f^{(4)}(\xi)}{4!}\int_a^b(x-a)\left(x-\frac{a+b}{2}\right)^2(x-b)\mathrm{d}x\\&=-\frac{f^{(4)}(\xi)}{2880}(b-a)^5\end{aligned}$$

9.2.9　复合 Simpson 积分公式及其截断误差

由 Simpson 积分公式的截断误差公式（9.2-6）可以看出，当积分区间较大时，截断误差可能会相当大，和梯形积分公式一样，复合 Simpson 积分公式才是实用公式.

复合 Simpson 积分公式推导过程如下.

将区间 $[a,b]$ 等分成 n 个子区间，节点编号为 $a=x_0,x_2,\cdots,x_{2n}=b$，$x_{2i}=a+2ih$，$h=\dfrac{b-a}{2n}$，$i=0,1,\cdots,n$. 现对子区间 $[x_{2(i-1)},\ x_{2i}]$，$i=1,2,\cdots,n$ 作 Simpson 积分，$L_2(x)$ 的节点取 $x_{2(i-1)}$，x_{2i-1}，x_{2i}，则

$$S_i=\int_{x_{2i-1}}^{x_{2i}}L_2^{(i)}(x)\mathrm{d}x=\frac{h}{3}f(x_{2i-1})+\frac{4}{3}hf(x_{2i-1})+\frac{h}{3}f(x_{2i}).$$

由此可得到复合 Simpson 积分公式为

$$\begin{aligned} S &= \sum_{i=1}^{n} S_i \\ &= \sum_{i=1}^{n} \int_{x_{2i-1}}^{x_{2i}} L_2^{(i)}(x)\mathrm{d}x \\ &= \left(\frac{1}{3} f(a) + \frac{1}{3} f(b) + \frac{2}{3} \sum_{i=1}^{n-1} f(x_{2i}) + \frac{4}{3} \sum_{i=1}^{n} f(x_{2i-1}) \right) h \end{aligned} \tag{9.2-7}$$

式（9.2-7）和一般教材稍有不同．不同之一是，式中 h 为子区间长度的一半，正因为这样，算式中的系数为其他书中的两倍；不同之二是，式（9.2-7）不出现半节点，这也是教材中取 $h = \dfrac{b-a}{2n}$ 的原因．

由复合 Simpson 积分公式与 Simpson 公式之间的关系，以及 Simpson 公式的截断误差，可直接得到复合 Simpson 积分公式的截断误差：

$$\begin{aligned} R(f) &= \int_a^b f(x)\mathrm{d}x - \sum_{i=1}^{n} \int_{x_{2i-1}}^{x_{2i}} L_2^{(i)}(x)\mathrm{d}x \\ &= \sum_{i=1}^{n} \int_{x_{2i-1}}^{x_{2i}} (f(x) - L_2^{(i)}(x))\mathrm{d}x \\ &= \sum_{i=1}^{n} \frac{-f^{(4)}(\xi_i)}{2880} (x_{2i} - x_{2i-2})^5 \\ &= \frac{-f^{(4)}(\xi)}{2880} n (x_{2i} - x_{2i-2})^5 \\ &= \frac{-f^{(4)}(\xi)}{2880} (b-a)(2h)^4 \\ &= -\frac{f^{(4)}(\xi)}{180} (b-a) h^4 \end{aligned} \tag{9.2-8}$$

比较式（9.2-4）和式（9.2-8）可知，复合 Simpson 积分公式的收敛性比复合梯形公式的收敛性好，计算精度高．

9.3 Newton 积分

n 阶 Newton 积分就是将节点间距离为 $h = \dfrac{b-a}{n}$ 的等距 Lagrange 插值多项式代替被积函数的积分公式．

9.3.1　通用 Newton 积分公式的求积系数和 Cotes 系数

$$\begin{aligned}\int_a^b L_n(x)\mathrm{d}x &= \int_a^b \sum_{i=0}^{n} l_i(x) f(x_i)\mathrm{d}x \\ &= \sum_{i=0}^{n} \int_a^b l_i(x) f(x_i)\mathrm{d}x \\ &= \sum_{i=0}^{n} A_i (b-a) f(x_i)\end{aligned} \tag{9.3-1}$$

$$\begin{aligned}A_i &= \int_a^b l_i(x)\mathrm{d}x \\ &= \int_a^b \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}\mathrm{d}x \\ &= h\int_0^n \frac{t(t-1)\cdots(t-i+1)(t-i-1)\cdots(t-n)}{(-1)^{n-1}t!(n-i)!}\mathrm{d}t \\ &= \frac{b-a}{n}\frac{(-1)^{n-1}}{i!(n-1)!}\int_0^n t(t-1)\cdots(t-i+1)(t-i-1)\cdots(t-n)\mathrm{d}t\end{aligned} \tag{9.3-2}$$

式中，A_i 为 n 阶 Newton 积分的求积系数，求积系数同积分区间有关．为了与不同积分区间有关，Cotes 计算出了各阶 Newton 积分的 Cotes 系数．

Cotes 系数和求积系数的关系为

$$C_i^{(n)} = A_i/(b-a)\,,\quad i = 0,1,\cdots,n \tag{9.3-3}$$

若用 Cotes 系数表示 Newton 积分，则

$$\int_a^b L_n(x)\mathrm{d}x = (b-a)\sum_{i=0}^{n} C_i^{(n)} f(x_i) \tag{9.3-4}$$

当 $n=1$（梯形积分）时，

$$C_0^{(1)} = -\int_0^1 (t-1)\mathrm{d}t = \frac{1}{2}\,,\quad C_1^{(1)} = -\int_0^1 t\mathrm{d}t = \frac{1}{2}\,.$$

于是

$$\int_a^b f(x)\mathrm{d}x = (b-a)\left(\frac{1}{2}f(a)+\frac{1}{2}f(b)\right).$$

这正是前面介绍的梯形积分公式．

当 $n=2$ 时，

$$C_0^{(2)} = \frac{1}{4}\int_0^2 (t-1)(t-2)\mathrm{d}t = \frac{1}{6}\,,\quad C_1^{(2)} = -\frac{1}{2}\int_0^2 t(t-2)\mathrm{d}t = \frac{2}{3}\,,\quad C_2^{(2)} = \frac{1}{4}\int_0^2 t(t-1)\mathrm{d}t = \frac{1}{6}\,.$$

于是

$$\int_a^b f(x)\mathrm{d}x=\frac{b-a}{6}\left(f(a)+4f(\frac{a+b}{2})+f(b)\right).$$

这正是前面介绍的 Simpson 积分公式.

Cotes 算出了各阶 Newton 积分的 Cotes 系数，详见表 9.4.

表 9.4 Cotes 系数表

n	$C_i^{(n)}$								
1	$\frac{1}{2}$	$\frac{1}{2}$							
2	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$						
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$					
4	$\frac{7}{90}$	$\frac{16}{45}$	$\frac{2}{15}$	$\frac{16}{45}$	$\frac{7}{90}$				
5	$\frac{19}{288}$	$\frac{25}{96}$	$\frac{25}{144}$	$\frac{25}{144}$	$\frac{25}{96}$	$\frac{19}{288}$			
6	$\frac{41}{840}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{34}{105}$	$\frac{9}{280}$	$\frac{9}{35}$	$\frac{41}{840}$		
7	$\frac{751}{17280}$	$\frac{3577}{17280}$	$\frac{1323}{17280}$	$\frac{2989}{17280}$	$\frac{2989}{17280}$	$\frac{1323}{17280}$	$\frac{3577}{17280}$	$\frac{751}{17280}$	
8	$\frac{989}{28350}$	$\frac{5888}{28350}$	$\frac{-928}{28350}$	$\frac{10496}{28350}$	$\frac{-4540}{28350}$	$\frac{10496}{28350}$	$\frac{-928}{28350}$	$\frac{5888}{28350}$	$\frac{989}{28350}$

9.3.2 *n* 阶 Newton 积分的代数精确度

前面已证明二阶 Newton 积分的代数精确度为 3．很容易证明梯形积分即一阶 Newton 积分的代数精确度为 1，现在证明任意阶的 Newton 积分的代数精确度.

【定理】 n 阶 Newton 积分的代数精确度 p 满足

$$p=\begin{cases}n+1, & n\text{为偶数}\\ n, & n\text{为奇数}\end{cases} \tag{9.3-5}$$

【证明】 当 n 为偶数时，设 $f(x)$ 是 $n+1$ 阶多项式，通过恒等变换，总可以将 $f(x)$ 转换成

$$f(x)=\sum_{i=0}^{n+1}\alpha_i\left(x-\frac{a+b}{2}\right)^i.$$

同样，经恒等变换总可以使 n 阶等距 Lagrange 插值多项式转换成

$$L_n(x)=\sum_{i=0}^{n}\beta_i\left(x-\frac{a+b}{2}\right)^i.$$

现将节点的编号改为

$$\begin{cases} x_0 = \dfrac{a+b}{2} \\ x_{2i-1} = x_0 + ih \\ x_{2i} = x_0 - ih \\ h = \dfrac{b-a}{n} \end{cases}, \quad i = 1,2,\cdots,\frac{n}{2}$$

显然，改变节点编号不会影响 $L_n(x)$ 的性质.

由代数插值多项式的性质

$$\begin{cases} \beta_0 = \alpha_0 \\ \sum\limits_{i=0}^{n} \beta_i (jh)^i = \sum\limits_{i=0}^{n+1} \alpha_i (jh)^i \\ \sum\limits_{i=0}^{n} \beta_i (-jh)^i = \sum\limits_{i=0}^{n+1} \alpha_i (-jh)^i \end{cases}, \quad j = 1,2,\cdots,\frac{n}{2} \tag{9.3-6}$$

将式（9.3-6）中奇序号方程和大一号的偶序号方程相加得

$$\beta_{2j} = \alpha_{2j}, \quad j = 1,2,\cdots,\frac{n}{2}$$

$$\int_a^b f(x)\mathrm{d}x = \int_a^b \sum_{i=0}^{n+1} \alpha_i \left(x - \frac{a+b}{2}\right)^i \mathrm{d}x = \int_a^b \sum_{i=0}^{n/2} \alpha_{2i} \left(x - \frac{a+b}{2}\right)^{2i} \mathrm{d}x,$$

$$\int_a^b L_n(x)\mathrm{d}x = \int_a^b \sum_{i=0}^{n} \beta_i \left(x - \frac{a+b}{2}\right)^i \mathrm{d}x = \int_a^b \sum_{i=0}^{n/2} \beta_{2i} \left(x - \frac{a+b}{2}\right)^{2i} \mathrm{d}x = \int_a^b f(x)\mathrm{d}x.$$

当 $f(x)$ 的阶为 $n+2$（$n+2$ 为偶数）时，$\alpha_{n+2} \neq 0$，$\beta_{n+2} = 0$，显然 $\int_a^b f(x)\mathrm{d}x \neq \int_a^b L_n(x)\mathrm{d}x$，所以代数精确度为 $n+1$.

当 n 为奇数时，若 $f(x)$ 是 n 阶多项式，由代数插值多项式的性质有 $L_n(x) = f(x)$，自然就有 $\int_a^b f(x)\mathrm{d}x = \int_a^b L_n(x)\mathrm{d}x$；当 $f(x)$ 是 $n+1$ 阶多项式时，$\alpha_{n+1} \neq 0$，而 $\beta_{n+1} = 0$，两者积分不等，这样定理就得到证明.

9.3.3　Newton 积分的截断误差

前面介绍了梯形积分公式和 Simpson 积分的截断误差，现给出一般 Newton 积分的截断误差. 当 $f(x)$ 在 $x \in [a,b]$ 上有足够高阶的连续导数时，n 阶 Newton 积分的截断误差为

$$\begin{aligned} R(f) &= \int_a^b f(x)\mathrm{d}x - \int_a^b L_n(x)\mathrm{d}x \\ &= \begin{cases} \displaystyle\int_a^b \frac{(a+b) f^{(n+2)}(\xi)}{(n+2)!} \left(x - \frac{b+a}{2}\right) \omega_n(x)\mathrm{d}x, & n\text{为偶数} \\ \displaystyle\int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_n(x)\mathrm{d}x, & n\text{为奇数} \end{cases} \end{aligned} \tag{9.3-7}$$

【证明】当 n 为偶数时，作 $n+1$ 阶带重节点的 Newton 插值多项式 $N_{n+1}(x)$，其节点为 a，$a+ih\,(i=1,2,\cdots,n)$，$h=\dfrac{b-a}{n}$，重节点为 $\dfrac{b-a}{2}$，显然 $L_n(x)$ 也是 $N_{n+1}(x)$ 的 n 阶插值多项式.

$$\int_a^b N_{n+1}(x)\mathrm{d}x=\int_a^b L_n(x)\mathrm{d}x$$

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x-\int_a^b L_n(x)\mathrm{d}x&=\int_a^b f(x)\mathrm{d}x-\int_a^b N_{n+1}(x)\mathrm{d}x\\&=\int_a^b\frac{(a+b)f^{(n+2)}(\xi)}{(n+2)!}\left(x-\frac{b+a}{2}\right)\omega_n(x)\mathrm{d}x\end{aligned}$$

当 n 为奇数时，可直接得到

$$\int_a^b f(x)\mathrm{d}x-\int_a^b L_n(x)\mathrm{d}x=\int_a^b\frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_n(x)\mathrm{d}x\ .$$

注意：重节点可任意取，当重节点不同时，ξ 值会不同. 对于 $f^{(n+1)}(\xi)$ 和 $f^{(n+2)}(\xi)$，当 $n>2$ 时不能提到积分号之外，因为 $\omega_n(x)$ 在 $[a,b]$ 上会变号，$\omega_n(x)\left(x-\dfrac{a+b}{2}\right)$ 也会变号.

9.3.4 Newton 积分的稳定性分析

Newton 积分的阶数愈高，代数精确愈高，截断误差愈小，但是当阶太高时会影响稳定性，计算误差会变大.

设 $C_i^{(n)}$ 是 n 阶 Newton 积分的第 i 个系数，所有 Newton 积分的代数精确度都大于等于 1，所以

$$\int_a^b 1\,\mathrm{d}x=(b-a)\sum_{i=0}^{n}C_i^{(n)}=b-a\ ,$$

即

$$\sum_{i=0}^{n}C_i^{(n)}=1\ .$$

假设 $f(x_i)$ 的测量值或计算值为 $\tilde{f}(x_i)$，有误差 ε_i，则其计算误差为

$$(b-a)\sum_{i=0}^{n}C_i^{(n)}f(x_i)-(b-a)\sum_{i=0}^{n}C_i^{(n)}\tilde{f}(x_i)=(b-a)\sum_{i=0}^{n}C_i^{(n)}\varepsilon_i\ .$$

若记 $\varepsilon=\max\left|\varepsilon_i\right|$，当 $C_i^{(n)}>0$ 时，

$$(b-a)\left|\sum_{i=0}^{n}C_i^{(n)}\varepsilon_i\right|\leqslant\left|b-a\right|\sum_{i=0}^{n}C_i^{(n)}\varepsilon=(b-a)\varepsilon\qquad(9.3\text{-}8)$$

也就是说，计算误差不大于 $(b-a)\varepsilon$，计算是稳定的. 当 $C_i^{(n)}$ 不全大于 0 时，式（9.3-8）不成立，算法的稳定性不保证.

从 Cotes 系数可以看出，$n>7$ 时 $C_i^{(n)}$ 有正有负，$n\leqslant 7$ 时 $C_i^{(n)}$ 全大于 0，故通常不用阶数

高于 7 的 Newton 积分.

9.4　Gauss 积分

Newton 积分以增加 $L_n(x)$ 的阶来提高代数精确度，n 阶 Newton 积分的代数精确度不超过 $n+1$，能否通过 $L_n(x)$ 的阶加上选择节点位置来提高代数精确度呢？Gauss 解决了这一问题.

9.4.1　选取节点位置和系数可提高代数精确度

对于数值积分公式 $\sum_{i=0}^{n} A_i f(x_i)$，可以通过选取 A_i 和 x_i 来提高求积分公式的代数精确度. 例如，对于积分

$$\int_0^1 f(x)\mathrm{d}x \approx A_0 f(x_0) + A_1 f(x_1)\,,$$

欲使求积公式 $A_0 f(x_0) + A_1 f(x_1)$ 的代数精确度为 3，须满足

$$\begin{cases} \int_0^1 1\,\mathrm{d}x = x\big|_0^1 = 1 = A_0 + A_1 \\ \int_0^1 x\mathrm{d}x = \dfrac{x^2}{2}\Big|_0^1 = \dfrac{1}{2} = A_0 x_0 + A_1 x_1 \\ \int_0^1 x^2\mathrm{d}x = \dfrac{x^3}{3}\Big|_0^1 = \dfrac{1}{3} = A_0 x_0^2 + A_1 x_1^2 \\ \int_0^1 x^3\mathrm{d}x = \dfrac{x^4}{4}\Big|_0^1 = \dfrac{1}{4} = A_0 x_0^3 + A_1 x_1^3 \end{cases}$$

解上述方程组得

$$\begin{cases} x_0 = \dfrac{3+\sqrt{3}}{6} \\ x_1 = \dfrac{3-\sqrt{3}}{6} \\ A_0 = \dfrac{1}{2} \\ A_1 = \dfrac{1}{2} \end{cases}$$

理论上讲，求积公式有 $n+1$ 个系数，$n+1$ 个节点，可提供 $2n+2$ 个条件，能使求积公式达到 $2n+1$ 阶代数精确度；但要解一个 $2n+2$ 阶非线性代数方程组，显然这一途径行不通.

9.4.2 正交多项式

虽然代数精确度为 $2n+1$ 的求积公式都称为 Gauss 积分，但真正有意义的 Gauss 积分是和正交多项式相关的．这里的正交多项式实际上是加权正交多项式．

【定义 1】 设 $\rho(x)$ 是定义在 $x\in[a,b]$ 上的连续函数，若满足

（1） $\rho(x)\geqslant 0$；

（2） $\int_a^b \rho(x)\mathrm{d}x>0$；

（3）积分 $\int_a^b x^n\rho(x)\mathrm{d}x$ 存在．

则称 $\rho(x)$ 为 $[a,b]$ 上的权函数．

【定义 2】 若 n 次多项式

$$g_n(x)=\sum_{i=0}^{n}a_i x^i\ ,\quad a_n\neq 0$$

满足

$$\int_a^b \rho(x)g_m(x)g_n(x)\mathrm{d}x=\begin{cases}0, & m\neq n\\ \int_a^b \rho(x)g_m^2(x)\mathrm{d}x>0, & m=n\end{cases}$$

则称多项式序列 $\{g_i(x)\}$ 在区间 $[a,b]$ 上带权 $\rho(x)$ 正交，并称 $g_n(x)$ 为区间 $[a,b]$ 上带权 $\rho(x)$ 的 n 次正交多项式．

【定理 1】 Legendre 多项式

$$\begin{cases}P_0(x)=1\\ P_n(x)=\dfrac{1}{2^n n!}\dfrac{\mathrm{d}^n((x^2-1)^n)}{\mathrm{d}^n x}, & n=1,2,\cdots\end{cases}$$

是以 1 为权函数的在区间 $[-1,1]$ 上的正交多项式．

证明略．

【定理 2】 Chebyshev 多项式

$$T_n(x)=\cos(n\arccos x)\ ,\quad n=1,2,\cdots$$

是区间 $[-1,1]$ 上带权 $\dfrac{1}{\sqrt{1-x^2}}$ 的正交多项式．

证明略．

下面列出一般正交多项式和 Gauss 积分的有关性质．

【性质 1】 任何次数不高于 n 的多项式 $q(x)$，可由正交多项式序列 $p_0(x),p_1(x),\cdots,p_n(x)$ 的线性组合表示，即

$$q(x)=c_0p_0(x)+c_1p_1(x)+\cdots+c_np_n(x)\ ,\ \text{系数}\ c_i\ \text{不全为 0}.$$

【性质 2】 任何在 $[a,b]$ 上带权 $\rho(x)$ 的正交多项式 $p_n(x)$，在 $[a,b]$ 上有 n 个不同的实根．

9.4.3　Gauss 积分

Gauss 积分也是用 n 阶 Lagrange 插值多项式代替 $f(x)$ 的数值积分. 和 Newton 积分不同，Newton 积分选取等距节点，而 Gauss 选取 $[a,b]$ 上带权 $\rho(x)$ 的 $n+1$ 阶正交多项式的零点为节点. 随所选正交多项式的不同，Gauss 积分分为 Gauss-Legendre 积分、Gauss-Chebyshev 积分……本教材只介绍 Gauss-Legendre 积分.

Gauss-Legendre 积分就是节点取 n+1 阶的 Legendre 多项式的零点的 n 次 Lagrange 插值多项式代替 $f(x)$ 的数值积分. Legendre 多项式的零点都在 $[-1,1]$ 之内，所以先考虑 $[-1,1]$ 上的 Gauss-Legendre 积分公式：

$$\int_{-1}^{1} L_n(x)\mathrm{d}x = \int_{-1}^{1}\sum_{i=0}^{n} l_i(x)f(x_i)\mathrm{d}x = \sum_{i=0}^{n} l_i(x)f(x_i)\mathrm{d}x = \sum_{i=0}^{n} g_i(x)f(x_i)\mathrm{d}x \tag{9.4-1}$$

式（9.4-1）中，x_i 就是 $n+1$ 阶 Legendre 多项式的零点，这些点称为 Gauss 点，g_i 称为 Gauss 系数. 当阶给定后，Gauss 点和 Gauss 系数早已被人工算出，可直接调用.

若积分区间为 $[a,b]$，则可通过坐标变换，将之映射到区域 $[-1,1]$ 上，因为

$$\int_a^b f(x)\mathrm{d}x = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b-a}{2}x+\frac{b+a}{2}\right)\mathrm{d}x\,,$$

所以在一般区间上的 Gauss-Legendre 计算公式为

$$\frac{b-a}{2}\int_{-1}^{1} L_n\left(\frac{b-a}{2}x_i+\frac{b+a}{2}\right)\mathrm{d}x = \frac{b-a}{2}\sum_{i=0}^{n} f\left(\frac{b-a}{2}x_i+\frac{b+a}{2}\right)g_i \tag{9.4-2}$$

式（9.4-2）中，x_i 仍是 $n+1$ 阶 Legendre 多项式的零点——Gauss 点，g_i 仍是 Gauss 系数，显然式（9.4-2）可代替式（9.4-1），是更通用的计算公式.

表 9.5 给出了 1 阶到 7 阶 Gauss-Legendre 积分的 Gauss 点和 Gauss 系数.

表 9.5　Gauss 点和 Gauss 系数

n	x_i	g_i	n	x_i	g_i
1	±0.55773503	1	5	±0.9324695142	0.1713244924
				±0.6612093865	0.3607615730
2	±0.7745967	0.5555556		±0.2386191861	0.4679139346
	0	0.8888889	6	±0.9491079123	0.1294849662
3	±0.8611363	0.3748548		±0.7415311856	0.2797053915
	±0.3398810	0.6521452		±0.4058451514	0.3818300505
				0	0.4179591837
4	±0.9061798	0.2369269	7	±0.9602898565	0.1012285363
	±0.5384963	0.4786287		±0.7966664774	0.2223810345
	0	0.5688889		±0.5255324099	0.3137066459
				±0.1834346425	0.3626837843

9.4.4 Gauss-Legendre 积分计算过程和计算实例

1．计算过程

（1）按所确定的 $L_n(x)$ 的阶，找出 Gauss 点和 Gauss 系数；

（2）计算 $f\left(\frac{b-a}{2}x_i+\frac{b+a}{2}\right)$，$i=0,1,\cdots,n$；

（3）按式（9.4-2）计算

$$G=\frac{b-a}{2}\sum_{i=0}^{n}g_i f\left(\frac{b-a}{2}x_i+\frac{a+b}{2}\right).$$

2．计算实例

【例 1】 分别用一阶和二阶 Gauss-Legendre 积分计算 $I=\int_{-1}^{1}\sqrt{x+1}\mathrm{d}x$.

【解】 计算结果见表 9.6 和表 9.7.

表 9.6 一阶 Gauss-Legendre 积分表

$\frac{b-a}{2}$	g_0	g_1	x_0	x_1	G
1	1	1	−0.57735027	0.57735027	1.90604123

表 9.7 二阶 Gauss-Legendre 积分表

$\frac{b-a}{2}$	g_0	g_1	g_2	x_0	x_1	x_2	G
1	0.5555556	0.8888889	0.5555556	−0.77459657	0	0.77459560	1.89272596

本例虽然积分区间为 $[-1,1]$，但表中仍给出 $\frac{b-a}{2}$，以示通用性.

9.4.5 *n* 阶 Gauss 积分代数精确度

【定理 1】 n 阶 Gauss 积分的代数精确度为 $2n+1$.

【证明】 设 $f(x)$ 是 $2n+1$ 阶多项式，因总能通过坐标变换使积分区间为 $[-1,1]$，因此我们只考虑积分 $\int_{-1}^{1}f(x)\mathrm{d}x$.

$$\begin{aligned}\int_{-1}^{1}f(x)\mathrm{d}x-\int_{-1}^{1}L_n(x)\mathrm{d}x&=\int_{-1}^{1}\big(f(x)-L_n(x)\big)\mathrm{d}x\\&=\int_{-1}^{1}(x-x_0)(x-x_1)\cdots(x-x_n)g(x)\mathrm{d}x\end{aligned}$$

式中，x_i（$i=0,1,\cdots,n$）为 $n+1$ 阶 Legendre 多项式的零点，因而 $(x-x_0)(x-x_1)\cdots(x-x_n)$ 是首系数为 1 的 $n+1$ 阶 Legendre 多项式，$g(x)$ 是 n 阶多项式，由正交多项式的性质有

$$g(x)=\sum_{i=0}^{n}\alpha_i p_i(x)，\ \alpha_i \text{ 不全为 } 0.$$

$p_i(x)$ 是 i 阶 Legendre 多项式，由正交多项式的正交性有

$$\int_{-1}^{1}(x-x_0)(x-x_1)\cdots(x-x_n)g(x)\mathrm{d}x=0\,.$$

当 $f(x)$ 是 $2n+2$ 阶多项式时，$g(x)$ 是 $n+1$ 阶多项式，此时有

$$g(x)=\sum_{i=0}^{n+1}\alpha_i p_i(x)\,,\quad \alpha_{n+1}\neq 0\,.$$

且有

$$\int_{-1}^{1}(x-x_0)(x-x_1)\cdots(x-x_n)g(x)\mathrm{d}x=\int_{-1}^{1}(x-x_0)(x-x_1)\cdots(x-x_n)\alpha_{n+1}p_{n+1}(x)\mathrm{d}x\neq 0\,.$$

也就是说，n 阶 Gauss-Legendre 积分的代数精确度为 $2n+1$．对于其他 Gauss 积分，也有同样的结果．

9.4.6　Gauss 积分的截断误差

n 阶 Newton 积分的截断误差实际上难以估计，因为

$$\int_a^b f^{(n+1)}(\xi)(x-x_0)(x-x_1)\cdots(x-x_n)\mathrm{d}x$$

通常是难以估计的，但 n 阶 Gauss 积分的截断误差较易估计．

【定理 2】若 $\int_a^b f(x)\mathrm{d}x$ 中 $f(x)$ 在区间 $[a,b]$ 上至少有 $2n+2$ 阶导数，则 n 阶 Gauss 积分的截断误差为

$$\frac{f^{(2n+2)}(\xi)}{(2n+2)!}(x-x_0)^2(x-x_1)^2\cdots(x-x_n)^2\,.$$

【证明】不妨仍假设 $a=-1$，$b=1$，作 $2n+1$ 阶带重节点的 Newton 插值多项式 $N_{2n}(x)$，节点 $x_0,x_1,\cdots,x_n$ 都是重节点，显然节点为 $x_0,x_1,\cdots,x_n$（都是 Gauss 点）的 $L_n(x)$ 也是 $2n+1$ 阶 Newton 插值多项式的插值多项式，由 Gauss 积分的代数精确度有

$$\int_{-1}^{1}L_n(x)\mathrm{d}x=\int_{-1}^{1}N_{2n+1}(x)\mathrm{d}x\,,$$

$$\begin{aligned}R(f)&=\int_{-1}^{1}f(x)\mathrm{d}x-\int_{-1}^{1}L_n(x)\mathrm{d}x\\&=\int_{-1}^{1}\big(f(x)-N_{2n+1}(x)\big)\mathrm{d}x\\&=\int_{-1}^{1}\frac{f^{(2n+2)}(\eta)}{(2n+2)!}(x-x_0)^2(x-x_1)^2\cdots(x-x_n)^2\mathrm{d}x\\&=\frac{f^{(2n+2)}(\xi)}{(2n+2)!}\int_{-1}^{1}(x-x_0)^2(x-x_1)^2\cdots(x-x_n)^2\mathrm{d}x\end{aligned}\,.$$

人们也常用 Gauss 复合积分公式来计算定积分．Gauss 复合积分公式的推导过程和 Newton 复合积分公式的推导过程相同，这里不再介绍．

9.4.7 Gauss 积分的稳定性和复合 Gauss 积分

由于 Gauss 系数都大于 0，仿 Newton 积分稳定性分析知，各阶 Gauss 积分都是稳定的.

9.5 Romberg 积分

Romberg 积分也称为 Romberg 算法，该算法系加速收敛法．Romberg 算法以复合梯形公式的逐次分半算法为基础，将复合梯形积分加权平均，利用外推法思想，构造出一种加速算法.

9.5.1 复合梯形积分公式逐次分半算法

复合梯形公式逐次分半算法是在原有计算的基础上，只计算新增节点相关值的一种算法.

将积分区间$[a,b]$分成n等份，按前面介绍的复合梯形公式计算T_n，需要计算$n+1$个函数值$f(x_i)$，$i=0,1,\cdots,n$．当T_n不满足精度要求时，将每个小区间分半，此时增加n个节点，利用复合梯形公式计算T_{2n}，由于T_{2n}中只有n个节点是新增的，因此T_{2n}和T_n之间有一定的关系：

$$\begin{aligned}
T_{2n} &= \frac{b-a}{4n}\left(f(a)+f(b)+2\sum_{i=1}^{2n-1} f\left(a+\frac{b-a}{2n}i\right)\right) \\
&= \frac{b-a}{4n}\left(f(a)+f(b)+2\sum_{i=1}^{n-1} f\left(a+\frac{b-a}{2n}2i\right)+2\sum_{i=1}^{n} f\left(a+\frac{b-a}{2n}(2i-1)\right)\right) \\
&= \frac{1}{2}T_n+\frac{b-a}{2n}\sum_{i=1}^{n} f\left(a+\frac{b-a}{2n}(2i-1)\right) \qquad (9.5\text{-}1)\\
&= \frac{1}{2}T_n+\frac{h_n}{2}\sum_{i=1}^{n} f\left(a+\frac{h_n}{2}(2i-1)\right) \\
&= \frac{1}{2}T_n+h_{2n}\sum_{i=1}^{n} f\left(a+(2i-1)h_{2n}\right)
\end{aligned}$$

为了便于编写程序，通常将区间分成2^k $(k=1,2,\cdots)$等份，设区间分成2^k等份时$h=h_k$．显然$h_{k+1}=\dfrac{h_k}{2}$，上式可归纳成

$$\begin{cases}
T_{2^0}=\dfrac{h_0}{2}(f(a)+f(b)) \\
T_{2^k}=\dfrac{1}{2}T_{2^{k-1}}+h_k\displaystyle\sum_{i=1}^{2^{k-1}} f(a+(2i-1)h_k) \\
h_k=\dfrac{h_{k-1}}{2},\quad h_0=b-a \\
\left|T_{2^k}-T_{2^{k-1}}\right|>\varepsilon
\end{cases}, \qquad k=1,2,\cdots,n \qquad (9.5\text{-}2)$$

当区间数以 2 倍的速度增长时，步长 h_k 以1/2的速度缩小，所以本算法也称为变步长梯形积分公式.

不少教材强调利用算法的截断误差估计区间数. 前面已经提到，这种方法理论上可行，实际上操作起来困难，原因是求一个数值函数的高阶导数最大值相当不易. 行之有效的办法是计算相邻结果差，由结果差决定是否还应该计算. 当然还有一种方法是加保险系数多算几次.

式（9.5-2）中不含半节点，因半节点手算虽然不太麻烦，但和计算语言中数组元素无对应关系，不便于编写程序.

9.5.2　复合梯形积分公式逐次分半算法的计算步骤

下面介绍复合梯形公式逐次分半算法的计算步骤.

（1）计算 $h=b-a$，$T_{2^0}=\dfrac{h}{2}(f(a)+f(b))$，即给出初始区间数 $n=1$；

（2）给出循环先导语句 $s=2*ep$；

（3）计算 T_k 和 s，计算公式为

$$T_{2^k}=\frac{1}{2}T_{2^{k-1}}+h_k\sum_{i=1}^{2^{k-1}}f(a+(2i-1)h_k)\ ,\quad k=1,2,\cdots,n$$

$$h_k=\frac{h_{k-1}}{2}$$

$s=\left|T_{2^k}-T_{2^{k-1}}\right|>\varepsilon$，$\varepsilon$ 为计算精度，即误差限

$$n\to n+n$$

9.5.3　复合梯形积分公式逐次分半算法的计算实例

【例 1】 用复合梯形公式逐次分半算法计算积分 $I=\int_0^1 \mathrm{e}^{-x}\mathrm{d}x$，要求 $\varepsilon<10^{-3}$.

【解】 按式（9.5-2）计算出的结果如表 9.8～表 9.12 所示.

表 9.8　梯形公式的计算结果

k	h	$f(0)$	$f(1)$	T_2^o
0	1	1	0.36788	0.68394

表 9.9　复合梯形公式逐次分半算法（$n=2$）的计算结果

k	h	$f\left(\frac{1}{2}\right)$	$\frac{T_{2^0}}{2}$	T_{2^1}
1	0.5	0.60653	0.34197	0.64524

表 9.10　复合梯形公式逐次分半算法（$n=4$）的计算结果

k	h	$f\left(\frac{1}{4}\right)$	$f\left(\frac{3}{4}\right)$	$\frac{T_{2^1}}{2}$	T_{2^2}
2	0.25	0.77880	0.47237	0.32262	0.63541

表 9.11 复合梯形公式逐次分半算法（$n=8$）的计算结果

k	h	$f\left(\frac{1}{8}\right)$	$f\left(\frac{3}{8}\right)$	$f\left(\frac{5}{8}\right)$	$f\left(\frac{7}{8}\right)$	$\frac{T_{2^2}}{2}$	T_{2^3}
3	0.125	0.88250	0.68729	0.53526	0.41686	0.36771	0.63294

表 9.12 复合梯形公式逐次分半算法（$n=16$）的计算结果

k	h	$f\left(\frac{1}{16}\right)$	$f\left(\frac{3}{16}\right)$	$f\left(\frac{5}{16}\right)$	$f\left(\frac{7}{16}\right)$	$f\left(\frac{9}{16}\right)$	$f\left(\frac{11}{16}\right)$	$f\left(\frac{13}{16}\right)$	$f\left(\frac{15}{16}\right)$	$\frac{T_{2^3}}{2}$	T_{2^4}
4	0.0625	0.93941	0.82903	0.73162	0.64565	0.56978	0.50283	0.44375	0.39161	0.31647	0.63233

由于 $\left|T_{2^4}-T_{2^3}\right|=0.61\times10^{-4}<\varepsilon=10^{-3}$，所以取 $I=0.63233$.

9.5.4 Romberg 积分公式

利用式（9.5-2）计算序列 $\{T_{2^k}\}$ 时，由于收敛速度不快，Romberg 给出了一个高速收敛算法.

由梯形公式可知，T_n 的截断误差与 h^2 成正比. 因此当步长缩小 1/2 时，截断误差约为原误差的 1/4，即

$$\frac{I-T_{2n}}{I-T_n}=\frac{1}{4},\quad I-T_{2n}\approx\frac{1}{3}(T_{2n}-T_n).$$

由此可见，当积分近似值 T_{2n} 与 T_n 相当接近时，就可以认为 T_{2n} 的误差很小，近似为 $(T_{2n}-T_n)/3$. 如果将误差加到 T_{2n} 上，则可以得到误差更小的近似值，即取

$$\tilde{T}=T_{2n}+\frac{1}{3}(T_{2n}-T_n)=\frac{4}{3}T_{2n}-\frac{1}{3}T_n=\frac{4T_{2n}-T_n}{4-1}.$$

下面推导出的式（9.5-3）的右端关系正好是复合梯形公式和复合 Simpson 公式之间的关系.

我们知道

$$T_n=\frac{h}{2}\left(f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right)$$

$$T_{2n}=\frac{h}{4}\left(f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)+2\sum_{i=1}^{n-1}f(x_{i-\frac{1}{2}})\right)$$

而 $x_i=a+ih$，$x_{i-\frac{1}{2}}=a+\left(i-\frac{1}{2}\right)h$，$h$ 是 T_n 的 h，即将积分区间分成 n 段的 h. 所以有

$$\frac{4T_{2n}-T_n}{4-1}=\frac{h\left(f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)+2\sum_{i=1}^{n-1}f(x_{i-\frac{1}{2}})\right)-\frac{h}{2}\left(f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right)}{3}$$

$$=\frac{h}{6}\left(f(a)+4\sum_{i=1}^{n}f(x_{i-\frac{1}{2}})+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right)$$

上式正是 Simpson 公式的另外一种表示法，即

$$S_n = \frac{T_{2n} - T_n}{4-1} \tag{9.5-3}$$

也就是说，在梯形公式序列 $\{T_{2^k}\}$ 的基础上，可以通过式（9.5-3）构成收敛速度快的 Simpson 序列 $\{S_{2^k}\}$.

同样，可在 Simpson 序列 $\{S_{2^k}\}$ 的基础上，通过式（9.5-4）构成收敛更快的新序列．这一序列称为 Cotes 序列 $\{C_{2^k}\}$：

$$C_n = \frac{16}{15}S_{2n} - \frac{1}{15}S_n = \frac{4^2 S_{2n} - S_n}{4^2 - 1} \tag{9.5-4}$$

在 Cotes 序列的基础上，还可构成比之收敛更快的 Romberg 序列 $\{R_{2^k}\}$：

$$R_n = \frac{4^3 C_{2n} - C_n}{4^3 - 1} \tag{9.5-5}$$

这种通过逐步构造收敛更快的序列来求积分近似值的算法，称为 Romberg 算法.

为了表述更简便，且利于编程，引入记号 $\{T_k^{(0)}\}$ 表示梯形序列，$\{T_k^{(1)}\}$ 表示 Simpson 序列……Romberg 算法可归纳成

$$\begin{cases} h_0 = b-a, \ \ h_i = \dfrac{h_{i-1}}{2} \\ T_{0,0} = \dfrac{h_0}{2}(f(a)+f(b)) \\ T_{i,0} = \dfrac{1}{2}T_{k-1}^{(0)} + h_i \displaystyle\sum_{k=1}^{2^{i-1}} f\left(a+(2k-1)h_i\right) & i = 1,2,\cdots,n \\ T_{i,j} = \dfrac{4^j T_{i,j-1} - T_{i-1,j-1}}{4^j - 1} & j = 1,2,\cdots,i \end{cases} \tag{9.5-6}$$

Romberg 计算过程如图 9.2 所示.

Romberg 积分计算精度相当高，通常计算到 $T_{4,4}$ 时精度足以满足要求，且式（9.5-6）的算法复杂性很低，可以一鼓作气算到 $T_{3,3}$ 或 $T_{4,4}$．用这种模式所给出算法和对应程序结构清晰，一个程序模块一个功能，程序易编不易出错.

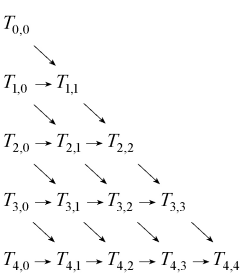

图 9.2　Romberg 计算过程

9.5.5　Romberg 积分公式的计算步骤

下面介绍 Romberg 积分公式的计算步骤.

（1）给出 $h = b - a$；

（2）用变步长梯形积分公式计算 $T_{i,0}$，计算公式为

$$T_{0,0}=\frac{h}{2}(f(a)+f(b))$$

$$T_{i,0}=\frac{1}{2}T_{i-1,0}+h\sum_{k=1}^{2^{i-1}}f(a+(2k-1)h)$$

（3）计算$T_{i,j}$，计算公式为

$$m_0=4$$

$$T_{i,j}=\frac{m_jT_{i,j-1}-T_{i-1,j-1}}{m_j-1},\quad i=1,2,\cdots,n,\quad j=1,2,\cdots,i$$

$$m_j=4m_{j-1}$$

式中 n=3 或 4.

9.5.6 Romberg 积分公式的计算实例

【例 2】用 Romberg 算法计算积分 $I=\int_0^1 \mathrm{e}^{-x}\mathrm{d}x$，要求 $\varepsilon<10^{-7}$.

【解】$T_{i,j}$ 的计算结果见表 9.13.

表 9.13 Romberg 积分的计算结果

$T_{i,j}$	$j=0$	$j=1$	$j=2$	$j=3$
$i=0$	0.68393972			
$i=1$	0.64523518	0.63233370		
$i=2$	0.63540941	0.63213414	0.63212085	
$i=3$	0.63294339	0.63212138	0.63212055	0.63212055

本例仅算到$T_{3,3}$，由结果可看出 Romberg 积分的计算精度很高.

9.6 导数数值算法

所有数值积分算法都是用函数的线性组合表示的，导数的数值算法也一样. 高等数学中，初等函数的导数、复合函数的导数、乘积函数的导数、反函数的导数以至它们的高阶导数，甚至多元函数的偏导数、高阶偏导数，都解决得非常好. 但所有算法都只宜于手算，难以编出通用程序. 对于数值计算而言，导数、高阶导数解决得不完美，即使对于一阶导数，一般教材和文献也未给出操作方便的能满足指定精度的通用算法.

目前，导数和高阶导数数值算法分成三类：差商法、外推法和利用代数插值多项式计算法.

9.6.1　差商法

差商法就是将微商（也就是导数）计算用差商代替．在高等数学中，导数的定义之一为

$$f'(a)=\lim_{h\to 0}\frac{f(a+h)-f(a)}{h}=\lim_{h\to 0}\frac{f(a)-f(a-h)}{h}=\lim_{h\to 0}\frac{f(a+h)-f(a-h)}{2h}$$

当 h 充分小时，按导数定义，可以用差商近似表示导数，从而得到几种数值微分公式：

$$f'(a)=\lim_{h\to 0}\frac{f(a+h)-f(h)}{h} \tag{9.6-1}$$

$$f'(a)=\lim_{h\to 0}\frac{f(a)-f(a-h)}{h} \tag{9.6-2}$$

$$f'(a)=\lim_{h\to 0}\frac{f(a+h)-f(a-h)}{2h} \tag{9.6-3}$$

式（9.6-1）、式（9.6-2）和式（9.6-3）分别称为向前差商、向后差商和中心差商．易证明：

$$f'(a)=\frac{f(a+h)-f(h)}{h}+O(h)$$

$$f'(a)=\frac{f(a)-f(a-h)}{h}+O(h)$$

$$f'(a)=\frac{f(a+h)-f(a-h)}{2h}+O(h^2)$$

从截断误差的角度看，h 应愈小愈好；但从稳定性出发 h 太小（分母接近 0）计算误差会很大，因而不宜太小．因此，差商法使用范围有限，计算实例见外推法算例．

9.6.2　外推法

前面已介绍过外推法，Romberg 积分法就属于外推法．若一个算法的截断误差为 $O(h^2)$，则可使用外推法．中心差商的截断误差为 $O(h^2)$，故可使用外推法．

对于中心差商，可将其表述成

$$\begin{cases} G_{00}(h)=\dfrac{f(a+h)-f(a-h)}{2h}=f'(a)+\alpha_1 h^2+\alpha_1 h^4+\cdots \\ G_{10}(\dfrac{h}{2})=\dfrac{f\left(a+\dfrac{h}{2}\right)-f\left(a-\dfrac{h}{2}\right)}{2\times\dfrac{h}{2}}=f'(a)+\dfrac{1}{4}\alpha_1 h^2+\dfrac{1}{16}\alpha_1 h^4+\cdots \end{cases} \tag{9.6-4}$$

仿 Romberg 积分算法做加权平均有

$$\begin{cases} G_{11}=\dfrac{4}{3}G_{10}\left(\dfrac{h}{2}\right)-\dfrac{1}{3}G_{00}(h) \\ G_{21}=\dfrac{16}{15}G_{11}\left(\dfrac{h}{2}\right)-\dfrac{1}{15}G_{10}(h) \\ \cdots \end{cases} \tag{9.6-5}$$

一般有

$$\begin{cases} G_{i,j}=\dfrac{4^j G_{i,j-1}-G_{i-1,j-1}}{4^j-1}=\dfrac{m_j G_{i,j-1}-G_{i-1,j-1}}{m_j-1}, & j=1,2,\cdots,n \\ m_j=4m_{j-1}, m_1=4, & i=j,j+1,\cdots,n \\ G_{i0}=\dfrac{f\left(a+\dfrac{h}{2i}\right)-f\left(a-\dfrac{h}{2i}\right)}{\dfrac{h}{2^{i-1}}}, & i=0,1,\cdots,n \end{cases} \tag{9.6-6}$$

通常 $i\leqslant 3$，否则稳定性不好.

9.6.3 外推法求导数计算步骤

外推法通常用来计算函数的一阶导数，其计算过程和 Romberg 积分的计算过程雷同，计算步骤如下.

1．用中心差商法计算 G_{i0}：

$$G_{i0}=\frac{f\left(a+\dfrac{h}{2i}\right)-f\left(a-\dfrac{h}{2i}\right)}{\dfrac{h}{2^{i-1}}},\quad i=0,1,2,3\ .$$

2．按式（9.6-6）计算 $G_{i,j}$，$j=1,2,3$；$i=j,j+1,\cdots,n$.

9.6.4 外推法计算实例

【例 1】用外推法计算 $G=(\mathrm{e}^x)'\big|_{x=1}$，$\varepsilon=10^{-6}$.

【解】计算公式见式（9.6-6），计算结果见表 9.14.

表 9.14 外推法计算导数表

i	h	G_{i0}	G_{i1}	G_{i2}	G_{i3}
0	0.8	3.01765			
1	0.4	2.79135	2.71592		
2	0.2	2.73644	2.718137	2.718285	
3	0.1	2.72281	2.718267	2.818276	2.718276

由表 9.14 可知，$(\mathrm{e}^x)'\big|_{x=1}=2.718276$.

9.6.5 利用插值多项式计算一阶导数和二阶导数

虽然插值多项式都是多项式，但只有幂级数插值多项式计算导数最方便．工程物理问题中常常要用一阶导数和二阶导数．这里只给出 4 阶幂级数型插值多项式及一阶导数和二阶导

数的计算公式.

1．4 阶幂级数型插值多项式

$$Z_4(x)=\sum_{i=0}^{4}\alpha_i(x-x^*)^i$$

下面是 $Z_4(x)$ 的生成步骤.

（1）节点选取如图 9.3 所示，节点坐标值如下：

$$\begin{cases}x_0=x^*=\dfrac{1}{2}(a+b)\\ h=\dfrac{b-a}{4}\\ x_1=x_0+h\\ x_3=x_0+2h\\ x_2=x_0-h\\ x_4=x_0-2h\end{cases}$$

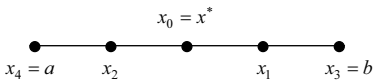

图 9.3　4 阶幂级数插值多项式节点图

对于 $y=f(x)$，$x\in[a,b]$，由插值多项式的性质有

$$\begin{cases}\alpha_0=y_0\\ \alpha_0+h\alpha_1+h^2\alpha_2+h^3\alpha_3+h^4\alpha_4=y_1\\ \alpha_0-h\alpha_1+h^2\alpha_2-h^3\alpha_3+h^4\alpha_4=y_2\\ \alpha_0+2h\alpha_1+4h^2\alpha_2+8h^3\alpha_3+16h^4\alpha_4=y_3\\ \alpha_0-2h\alpha_1+4h^2\alpha_2-8h^3\alpha_3+16h^4\alpha_4=y_4\end{cases}$$

解之得

$$\begin{cases}\alpha_0=y_0\\ \alpha_3=\dfrac{2y_2-2y_1+y_3-y_4}{12h^3}\\ \alpha_1=\dfrac{y_1-y_2-2h^3\alpha_3}{2h}\\ \alpha_4=\dfrac{y_3+y_4-4y_1-4y_2+6\alpha_0}{24h^4}\\ \alpha_2=\dfrac{y_1+y_2-2h^4\alpha_4-2\alpha_0}{2h^2}\end{cases}\tag{9.6-7}$$

2．一阶导数的计算公式

$$Z_4'(x)=\alpha_1+2\alpha_2(x-x^*)+3\alpha_3(x-x^*)^2+4\alpha_4(x-x^*)^3\tag{9.6-8}$$

3．二阶导数的计算公式

$$Z_4''(x)=2\alpha_2+6\alpha_3(x-x^*)+12\alpha_4(x-x^*)^2\tag{9.6-9}$$

9.6.6 用幂级数型插值多项式计算函数的一阶和二阶导数算例及计算过程

【例 2】用式（9.6-8）、式（9.6-9）计算 $y=\sqrt{1+x}$，$x\in[0,1]$ 在 $x=0.3$ 处的一阶和二阶导数.

【解】（1）给出 $y=\sqrt{1+x}$，$x\in[0,1]$ 的节点数据，取 $h=0.25$，$x^*=0$.

i	0	1	2	3	4
x_i	0.5	0.75	0.25	1	0
y_i	1.224745	1.322876	1.118034	1.414214	1.000000

（2）按式（9.6-7）计算 $Z_4(x)=\sum_{i=0}^{4}\alpha_i(x-x^*)^i$ 的系数：

$$\alpha_0=y_0=1.224745$$

$$\alpha_4=\frac{y_3+y_4-4y_1-4y_2+6\alpha_0}{24h^4}=-0.010192$$

$$\alpha_2=\frac{y_1+y_2-2h^4\alpha_4-2\alpha_0}{2h^2}=-0.068005$$

$$\alpha_3=\frac{2y_2-2y_1+y_3-y_4}{12h^3}=0.024162$$

$$\alpha_1=\frac{y_1-y_2-2h^3\alpha_3}{2h}=0.408173$$

（3）按式（9.6-8）和式（9.6-9）计算一阶和二阶导数值：

$$Z_4'(0.3)=\alpha_1+2\alpha_2x+3\alpha_3x^2+4\alpha_4x^3=0.438601$$

$$Z_4''(0.3)=2\alpha_2+6\alpha_3x+12\alpha_4x^2=-0.169895$$

习题 9

1．说明积分公式

$$\int_a^b f(x)\mathrm{d}x\approx(b-a)f\left(\frac{a+b}{2}\right)$$

的几何意义，并证明

$$\int_a^b f(x)\mathrm{d}x=(b-a)f\left(\frac{a+b}{2}\right)+\frac{(b-a)^3}{24}f''(\eta)，\quad \eta\in[a,b].$$

2．分别用梯形公式、Simpson 公式计算积分 $\int_0^1 \mathrm{e}^{-x}\mathrm{d}x$ 的近似值，并估计误差.

3．将区间 [0,1] 分成 4 等份，分别用复合梯形公式、复合 Simpson 公式计算积分 $\int_0^1 \mathrm{e}^{-x}\mathrm{d}x$ 的近似值.

4．试判定如下求积公式的代数精确度：

$$\int_0^1 f(x)\mathrm{d}x\approx\frac{3}{4}f\left(\frac{1}{3}\right)+\frac{1}{4}f(1).$$

5．确定 A_{-1}，A_0 和 A_1，使如下积分公式的代数精确度尽量高，并指明求积公式所具有的代数精确度：

$$\int_{-2h}^{2h} f(x)\mathrm{d}x \approx A_{-1}f(-h) + A_0 f(0) + A_1 f(h)\ .$$

6．确定 x_1，x_2，使如下积分公式的代数精确度尽量高，并指明求积公式所具有的代数精确度：

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{1}{3}(f(-1) + 2f(x_1) + 3f(x_2))\ .$$

7．用 Romberg 积分法计算积分 $\int_0^1 \mathrm{e}^x \mathrm{d}x$，精确到 10^{-5}．

8．确定下列 Gauss 型求积公式中的系数和节点：

（1） $\int_0^1 \ln x f(x)\mathrm{d}x \approx A_1 f(x_1) + A_2 f(x_2)$；

（2） $\int_0^1 \frac{1}{\sqrt{x}} f(x)\mathrm{d}x \approx A_1 f(x_1) + A_2 f(x_2)$．

9．分别用中心差商（取 $h = 0.1$）和外推法（取 $i = 3$）计算 $f(x) = \ln(1+x)$ 在 $x = 1.5$ 处的一阶导数值．

10.用 4 阶幂级数型插值多项式计算 $f(x) = \ln(1+x)$，$x \in [1,3]$ 在 $x = 1.5$ 处的一阶和二阶导数值．

第 10 章　常微分方程初值问题的数值解

自然界的很多现象，例如生物的生灭过程、物体的运动、化学反应过程等，就其量的变化规律而言，不少可以归纳为以时间 t 为自变量的常微分方程（组）. 但是，只有极少数的典型常微分方程可以求出其理论解，大部分问题只能用数值方法来求其近似解. 本章只讨论一阶常微分方程初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = f(t,y), & 0 \leqslant t \leqslant T \\ y(0) = y_0 \end{cases} \tag{10.0-1}$$

的数值解.

在讨论数值解之前，需要给出一些定义及与常微分方程理论相关的一些结论.

【定义 1】 若函数 $f(t,y)$ 在集合 $D \subset \boldsymbol{R}^2$ 中满足以下不等式：

$$\left|f(t,y_1) - f(t,y_2)\right| \leqslant L\left|y_1 - y_2\right| \tag{10.0-2}$$

其中常数 $L > 0$ 且任意 (t,y_1)，$(t,y_2) \in D$，则称 $f(t,y)$ 关于变量 y 满足 Lipschitz 条件，常数 L 称为 Lipschitz 常数.

【定义 2】 集合 $D \subset \boldsymbol{R}^2$，若 (t,y_1)，$(t,y_2) \in D$，且当 $0 \leqslant \lambda \leqslant 1$ 时，对任意 λ，有

$$((1-\lambda)t_1 + \lambda t_2, (1-\lambda)y_1 + \lambda y_2) \in D,$$

则称 D 是凸集.

易证：集合 $D = \{(t,y) \mid a \leqslant t \leqslant b,\ -\infty \leqslant y \leqslant +\infty\}$（其中 a 和 b 为常数）是凸集.

【定理 1】 假设函数 $f(t,y)$ 在凸区域 $D = \{(t,y) \mid a \leqslant t \leqslant b,\ -\infty \leqslant y \leqslant +\infty\}$ 上连续，并且关于 y 满足 Lipschitz 条件，则常微分方程初值问题式（10.0-1）的解存在且唯一，并且还连续依赖于初值 $y(0)$.

证明略.

【定义 3】 若初值问题［即式（10.0-1）］满足下列条件，则称初值问题是适定的：

（1）解存在且唯一；

（2）存在 $\varepsilon > 0$，扰动问题

$$\begin{cases} \dfrac{\mathrm{d}z}{\mathrm{d}t} = f(t,z) + \delta(z), & a \leqslant t \leqslant b \left|\delta(z)\right| < \varepsilon \\ z(0) = y_0 + \varepsilon_0 \end{cases}$$

的解存在且唯一；

（3）存在常数 k，使

$$\left|z(t) - y(t)\right| < k\varepsilon_0, \qquad 0 \leqslant t \leqslant T.$$

【定理 2】对于初值问题［即式（10.0-1）］，如果 f 在凸区域 G 上连续，且关于 y 满足 Lipschitz 条件，则初值问题是适定的.

证明略.

初值问题的数值解法就是求函数 $y(t)$ 在区间 $[0,T]$ 内一系列节点 $0=t_0<t_1<\cdots<t_n=T$ 处理论值 $y(t_i)$ 的近似值 y_i $(i=0,1,\cdots,n)$. 通常考虑等距节点的情形：$t_i=ih$，其中 $h=\dfrac{T}{n}$ 称为步长.

下面逐一介绍几种常用的数值解法.

10.1　Euler 法

Euler 法在实际计算中用得不多，但因其格式十分简单，可以用来说明其他较高级方法中的推导技巧，而避免烦琐的代数问题.

10.1.1　Euler 公式的推导

对于一阶常微分方程的初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}t}=f(t,y), & 0\leqslant t\leqslant T\\ y(0)=y_0\end{cases} \tag{10.1-1}$$

假设式（10.1-1）的唯一解 $y(t)$ 在 $[0,T]$ 上有二阶连续导数，则对 $i=0,1,\cdots,n-1$，利用 Taylor 展开式得

$$y(t_{i+1})=y(t_i)+(t_{i+1}-t_i)y'(t_i)+\frac{(t_{i+1}-t_i)^2}{2!}y''(\xi_i),$$

其中 $t_i<\xi_i<t_{i+1}$，$i=0,1,\cdots,n-1$.

因为 $y(t)$ 为微分方程式（10.1-1）的解，所以将 $y'(t_i)=f(t_i,y(t_i))$ 代入上面的 Taylor 展开式，再利用节点等距，得

$$y(t_{i+1})=y(t_i)+hy'(t_i)+\frac{h^2}{2!}y''(\xi_i) \tag{10.1-2}$$

当 h 充分小时，略去高阶无穷小量 $\dfrac{h^2}{2!}y''(\xi_i)$，得

$$y(t_{i+1})\approx y(t_i)+hy'(t_i).$$

所谓 Euler 法，是指分别用 y_{i+1} 和 y_i 作为 $y(t_{i+1})$ 和 $y(t_i)$ 的近似值（$i=0,1,\cdots,n-1$），并且满足递推关系

$$\begin{cases}y_0=y(0)\\ y_{i+1}=y_i+hf(t_i,y_i)\end{cases} \tag{10.1-3}$$

的数值算法，称式（10.1-3）为 Euler 法的差分格式，或者就称式（10.1-3）为 Euler 法.

10.1.2 Euler 法的计算步骤

Euler 法的计算过程如下.

（1）计算步长 h 和各时刻 t_i 的值.

计算公式为

$$h=\frac{T}{n},\quad t_i=ih,\quad i=0,1,\cdots,n$$

（2）按式（10.1-3）计算 y_i 值.

计算公式为

$$y_{i+1}=y_i+hf(t_i,y_i),\quad i=0,1,\cdots,n-1$$

【例 1】 用 Euler 法求解初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}t}=ty, & 0\leqslant t\leqslant 1\\ y(0)=1\end{cases}$$

的近似解.

【解】 取 $n=10$，$h=\dfrac{1}{10}$，按公式

$$\begin{cases}y_0=1\\ y_i=y_{i-1}+hf(t_{i-1},y_{i-1})=y_{i-1}+0.1t_{i-1}y_{i-1}, & i=1,2,\cdots,10\end{cases}$$

计算，计算结果如表 10.1 所示.

此例的理论解为 $y(t)=\mathrm{e}^{\frac{t^2}{2}}$，表 10.1 中的 $y(t_i)$ 为理论解.

表 10.1 Euler 法解微分方程的计算结果

i	t_i	y_i	$y(t_i)$	$\lvert y_i-y(t_i)\rvert$
0	0.0	1.000000000	1.000000000	0.000000000
1	0.1	1.000000000	1.005012521	0.005012521
2	0.2	1.010000000	1.020201340	0.010201340
3	0.3	1.030200000	1.046027860	0.015827860
4	0.4	1.061106000	1.083287068	0.022181068
5	0.5	1.103550240	1.133148453	0.029598213
6	0.6	1.158727752	1.197217363	0.038489611
7	0.7	1.228251417	1.277621313	0.049369896
8	0.8	1.314229016	1.377127764	0.062898748
9	0.9	1.419367338	1.499302500	0.079935162
10	1.0	1.547110398	1.648721271	0.101610873

由表 10.1 可知，误差 $\lvert y_i-y(t_i)\rvert$ 随 t_i 的增加而增大.

10.1.3　Euler 法的截断误差

先讨论 Euler 法的局部截断误差．局部截断误差是指在假定第 i 步的理论值 $y(t_i)$ 和由差分格式求出的近似值 y_i 相等的前提下，第 $i+1$ 步的理论值 $y(t_{i+1})$ 与近似值 y_{i+1} 的差，即当 $y(t_i)=y_i$ 时，局部截断误差为

$$\varepsilon_{i+1}=y(t_{i+1})-y_{i+1}\,.$$

下面导出 Euler 法局部截断误差 ε_{i+1} 的表达式．

将式（10.1-1）两端积分，得

$$y(t_{i+1})-y_{i+1}=\int_{t_i}^{t_{i+1}}f(t,y(t))\mathrm{d}t\,,$$

将此式减去式（10.1-3）得

$$\varepsilon_{i+1}=y(t_{i+1})-y_{i+1}=y(t_i)-y_i+\int_{t_i}^{t_{i+1}}f(t,y(t))\mathrm{d}t-hf(t_i,y_i)\tag{10.1-4}$$

由局部截断误差假设 $y(t_i)=y_i$，得

$$\varepsilon_{i+1}=y(t_{i+1})-y_{i+1}=\int_{t_i}^{t_{i+1}}f(t,y(t))\mathrm{d}t-hf(t_i,y_i)\tag{10.1-5}$$

关于 Euler 法的局部截断误差有下述定理．

【定理 1】若函数 $f(t,y)$ 在凸区域 $D=\{(t,y)\,|\,a\leqslant t\leqslant b,\ -\infty\leqslant y\leqslant+\infty\}$ 上关于变量 t，y 都满足 Lipschitz 条件，即

$$\left|f(t_1,y)-f(t_2,y)\right|\leqslant K\left|t_1-t_2\right|\tag{10.1-6}$$

$$\left|f(t,y_1)-f(t,y_2)\right|\leqslant L\left|y_1-y_2\right|\tag{10.1-7}$$

则 Euler 法的局部截断误差满足

$$\left|\varepsilon_{i+1}\right|\leqslant\frac{h^2}{2}(K+LM)\,,\quad i=0,1,\cdots,n-1\tag{10.1-8}$$

其中，$M=\max\limits_{0\leqslant t\leqslant T}\left|f(t,y(t))\right|$．

【证明】由式（10.1-5）～式（10.1-8）及微分中值定理得

$$\begin{aligned}
\left|\varepsilon_{i+1}\right|&=\left|\int_{t_i}^{t_{i+1}}f(t,y(t))\mathrm{d}t-hf(t_i,y(t_i))\right|\\
&=\left|\int_{t_i}^{t_{i+1}}f(t,y(t))\mathrm{d}t-\int_{t_i}^{t_{i+1}}f(t_i,y(t_i))\mathrm{d}t\right|\\
&=\left|\int_{t_i}^{t_{i+1}}((f(t,y(t))-f(t_i,y(t)))+(f(t_i,y(t))-f(t_i,y(t_i)))\mathrm{d}t\right|\\
&\leqslant\int_{t_i}^{t_{i+1}}\left|f(t,y(t))-f(t_i,y(t))\right|\mathrm{d}t+\int_{t_i}^{t_{i+1}}\left|f(t_i,y(t))-f(t_i,y(t_i))\right|\mathrm{d}t\\
&\leqslant K\int_{t_i}^{t_{i+1}}\left|t-t_i\right|\mathrm{d}t+L\int_{t_i}^{t_{i+1}}\left|y(t)-y(t_i)\right|\mathrm{d}t\\
&\leqslant\frac{Kh^2}{2}+L\int_{t_i}^{t_{i+1}}\left|y'(\xi_t)(t-t_i)\right|\mathrm{d}t
\end{aligned}$$

设 $M=\max\limits_{a\leqslant t\leqslant T}\left|y'(t)\right|=\max\limits_{a\leqslant t\leqslant T}\left|f(t,y(t))\right|$，则

$$\left|\varepsilon_{i+1}\right|\leqslant\frac{h^2}{2}(K+LM)\,.$$

上式表明，当 h 充分小时，Euler 法的局部截断误差为 $O(h^2)$.

下面再讨论 Euler 法的整体截断误差. 所谓整体截断误差，是指由初始值 y_0 出发，在没有舍入误差的前提下，由差分递推关系式（10.1-3）求出近似值 y_i 与理论值 $y(t_i)$ 之间的误差，即整体误差

$$E_i=y(t_i)-y_i,\ \ i=0,1,\cdots,n\,.$$

在给出 Euler 法的整体截断误差之前，先给出两个引理，它们是建立 Euler 法整体截断误差的计算依据.

【引理 1】 对任意 $x>0$ 及任意正数 n，有

$$(1+x)^n<\mathrm{e}^{nx} \tag{10.1-9}$$

证明略.

【引理 2】 设 $s>0$，t 为非负实数，序列 $\{a_i\}$ 满足递推不等式

$$\left|a_{i+1}\right|\leqslant(1+s)\left|a_i\right|+t \tag{10.1-10}$$

则

$$\left|a_{i+1}\right|\leqslant\mathrm{e}^{(i+1)s}\left(\frac{t}{s}+\left|a_0\right|\right)-\frac{t}{s} \tag{10.1-11}$$

证明略.

【定理 2】 若函数 $f(t,y)$ 在凸区域 $D=\{(t,y)\,|\,a\leqslant t\leqslant b,\ -\infty\leqslant y\leqslant+\infty\}$ 上关于变量 y 满足 Lipschitz 条件，即

$$\left|f(t,y_1)-f(t,y_2)\right|\leqslant L\left|y_1-y_2\right|,$$

则 Euler 法的整体截断误差为

$$\left|\varepsilon_{i+1}\right|\leqslant\frac{hM}{2L}(\mathrm{e}^{L(b-a)}-1) \tag{10.1-12}$$

其中，$M=\max\limits_{a\leqslant t\leqslant T}\left|y''(t)\right|$.

【证明】 将式（10.1-2）的 $y(k_{i+1})$ 代入式（10.1-4），得

$$\left|E_{i+1}\right|=\left|y(t_{i+1})-y_{i+1}\right|=\left|y(t_i)-y_i+h\big(f(t_i,y(t_i))-f(t_i,y_i)\big)+\frac{h^2}{2}y''(\xi_i)\right|,$$

利用 $f(t,y)$ 关于变量 y 满足 Lipschitz 条件，并且 $\left|y''(t)\right|\leqslant M$，得

$$\left|E_{i+1}\right|\leqslant\left|E_i\right|+hL\left|y(t_i)-y_i\right|+\frac{h^2}{2}\left|y''(\xi_i)\right|\leqslant(1+hL)\left|E_i\right|+\frac{h^2}{2}M\,,$$

利用引理 2，得

$$|E_{i+1}| \leqslant \mathrm{e}^{(i+1)hL}\left(\frac{h^2M}{2Lh}+|E_0|\right)-\frac{h^2M}{2Lh}.$$

由于 $E_0 = y(t_0)-y_0 = 0$，从而有

$$|E_{i+1}| \leqslant \frac{hM}{2L}(\mathrm{e}^{(i+1)hL}-1),$$

因为 $(i+1) \leqslant n$，$h=\dfrac{b-a}{n}$，所以

$$|E_{i+1}| \leqslant \frac{hM}{2L}(\mathrm{e}^{L(b-a)}-1).$$

式（10.1-12）表明，当 Euler 法不考虑初值误差时，整体截断误差 $E_{i+1}=O(h)$，即整体截断误差比局部截断误差低一阶．这一结论带有一般性．

【定义】常微分方程初值问题的数值计算方法的精度等于该算法整体截断误差的阶．

由定理 2 可知，Euler 法是精度为 1 的数值方法．

10.2　改进 Euler 法和预估-校正法

用 Euler 法解常微分方程初值问题，尽管其运算简单，但其局部截断误差的阶仅为 2，整体截断误差的阶仅为 1．现在考虑在 Euler 法的基础上，建立新的数值解法，以便使其局部截断误差的阶有所提高．

10.2.1　改进 Euler 法

已知初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}t}=f(t,y), & 0 \leqslant t \leqslant T\\ y(0)=y_0\end{cases} \tag{10.2-1}$$

微分方程式（10.2-1）等价于以下积分方程：

$$y(t_2)-y(t_1)=\int_{t_1}^{t_2} f(t,y(t))\mathrm{d}t \tag{10.2-2}$$

对方程式（10.2-2）右端的积分采用梯形公式，并令 $t_1=t_i$，$t_2=t_{i+1}$ 且 $h=t_{i+1}-t_i$，得

$$y(t_{i+1})=y(t_i)+\frac{h}{2}\big(f(t_i,y(t_i))+f(t_{i+1},y(t_{i+1}))\big)+R_{i+1} \tag{10.2-3}$$

其中，

$$\begin{aligned}R_{i+1}&=\int_{t_i}^{t_{i+1}} f(t,y)\mathrm{d}t-\frac{h}{2}\big(f(t_i,y(t_i))+f(t_{i+1},y(t_{i+1}))\big)\\&=\int_{t_i}^{t_{i+1}} y'(t)\mathrm{d}t-\frac{h}{2}\big(y'(t_i)+y'(t_{i+1})\big)\\&=-\frac{h^3}{12}y'''(\xi_i), \qquad \xi_i\in[t_i,t_{i+1}]\end{aligned}$$

当 h 充分小时，略去无穷小量 $-\frac{h^3}{12}y'''(\xi_i)$，得

$$y(t_{i+1}) = y(t_i) + \frac{h}{2}\big(f(t_i, y(t_i)) + f(t_{i+1}, y(t_{i+1}))\big).$$

所谓改进 Euler 法，是指分别用 y_{i+1}，y_i 作为 $y(t_{i+1})$，$y(t_i)$ 的近似值，满足递推关系式

$$y_{i+1} = y_i + \frac{h}{2}\big(f(t_i, y_i) + f(t_{i+1}, y_{i+1})\big) \tag{10.2-4}$$

的数值算法.

【定理 1】对于初值问题式（10.2-1），设 $f(t,y)$ 及 $f_y = f'_y(t,y)$ 在凸区域

$$D = \{(t,y) \mid a \leqslant t \leqslant b,\ -\infty \leqslant y \leqslant +\infty\}$$

内连续，且当 $\frac{h}{2}|f_y| < 1$ 时，改进 Euler 法的局部截断误差满足

$$|\varepsilon_{i+1}| \leqslant \frac{h^3 M}{12\left(1 - \frac{h}{2}|f_y|\right)},\quad i = 0,1,\cdots,n-1 \tag{10.2-5}$$

其中，$M = \max\limits_{t_i \leqslant \xi_i \leqslant t_{i+1}} |y'''(\xi_i)|$，$i = 0,1,\cdots,n-1$.

【证明】在局部截断误差的前提（$y(t_i) = y_i$）下，由式（10.2-3）减去式（10.2-4），并运用微分中值定理，得

$$\begin{aligned}|\varepsilon_{i+1}| &= \left|\frac{h}{2}\big(f(t_{i+1}, y(t_{i+1})) - f(t_{i+1}, y_{i+1})\big) - \frac{h^3}{12}y'''(\xi_i)\right| \\ &\leqslant \frac{h}{2}|f_y||\varepsilon_{i+1}| + \frac{h^3}{12}M,\end{aligned}$$

从而当 $\frac{h}{2}|f_y| < 1$ 时，有

$$|\varepsilon_{i+1}| \leqslant \frac{h^3 M}{12\left(1 - \frac{h}{2}|f_y|\right)}.$$

定理 1 表明，改进 Euler 法的局部截断误差阶为 $O(h^3)$，它比 Euler 法的局部截断误差高一阶.

比较 10.1 节中的 Euler 法［即式（10.1-4）］和本节的改进 Euler 法［即式（10.2-4）］，可以看出在求解 y_{i+1} 时的一个重要差别．Euler 法在已知 y_i 时，可由递推公式

$$y_{i+1} = y_i + hf(t_i, y_i)$$

直接计算出 $y_{i+1}(i = 0,1,\cdots,n-1)$，而不必解任何方程．这类方法称为显式法．10.1 节中的计算公式（10.1-4）为显式 Euler 法．本节的改进 Euler 法，在已知 y_i 的情况下，求解 y_{i+1} 需要解方程式（10.2-4），这类方法称为隐式法．计算公式（10.2-4）也称为隐式 Euler 法.

从计算角度讲，显式法比隐式法易于求解，隐式法往往采用迭代法求解，但隐式法的精度比同阶显式法的精度高，稳定性好.

10.2.2　改进 Euler 法的收敛性

先考虑方程式（10.2-4）的多次迭代解法．取

$$y_{i+1}^{(0)} = y_i + hf(t_i, y_i) \tag{10.2-6}$$

作为初值；用迭代公式

$$y_{i+1}^{(k+1)} = y_i + \frac{h}{2}(f(t_i, y_i) + f(t_{i+1}, y_{i+1}^{(k)})),\quad k = 0,1,\cdots \tag{10.2-7}$$

求出迭代序列

$$y_{i+1}^{(0)},\ y_{i+1}^{(1)},\ \cdots,\ y_{i+1}^{(k)},\cdots \tag{10.2-8}$$

在一定条件下，序列式（10.2-8）收敛于方程式（10.2-1）的解 y_{i+1}．

【定理 2】如果函数 $f(t,y)$ 在凸区域 $D=\{(t,y)\,|\,a\leqslant t\leqslant b,\ -\infty\leqslant y\leqslant+\infty\}$ 上关于变量 y 满足 Lipschitz 条件，即

$$\left|f(t,y_1)-f(t,y_2)\right|\leqslant L\left|y_1-y_2\right|,$$

则当 $\frac{1}{2}hL<1$ 时，由迭代公式（10.2-6）和式（10.2-7）求出的序列式（10.2-8）收敛于方程式（10.2-4）的解 y_{i+1}．

【证明】只需证明由迭代公式（10.2-6）和式（10.2-7）求出的序列式（10.2-8）为一个 Cauchy 序列．

由式（10.2-7）得

$$\begin{aligned}\left|y_{i+1}^{(k+1)}-y_{i+1}^{(k)}\right| &= \left|y_i+\frac{h}{2}(f(t_i,y_i)+f(t_{i+1},y_{i+1}^{(k)}))-y_i-\frac{h}{2}(f(t_i,y_i)+f(t_{i+1},y_{i+1}^{(k-1)}))\right| \\ &= \frac{h}{2}\left|f(t_{i+1},y_{i+1}^{(k)})-f(t_{i+1},y_{i+1}^{(k-1)})\right| \\ &\leqslant \frac{1}{2}hL\left|y_{i+1}^{(k)}-y_{i+1}^{(k-1)}\right| \\ &\leqslant \left(\frac{1}{2}hL\right)^k\left|y_{i+1}^{(1)}-y_{i+1}^{(0)}\right|\end{aligned}$$

从而对任意 m，n（$m>n$），有

$$\begin{aligned}\left|y_{i+1}^{(m)}-y_{i+1}^{(n)}\right| &\leqslant \left|y_{i+1}^{(m)}-y_{i+1}^{(m-1)}\right|+\left|y_{i+1}^{(m-1)}-y_{i+1}^{(m-2)}\right|+\cdots+\left|y_{i+1}^{(n+1)}-y_{i+1}^{(n)}\right| \\ &\leqslant \left[\left(\frac{1}{2}hL\right)^{m-1}+\left(\frac{1}{2}hL\right)^{m-2}+\cdots+\left(\frac{1}{2}hL\right)^{n}\right]\left|y_{i+1}^{(1)}-y_{i+1}^{(0)}\right| \\ &= \frac{\left(\frac{1}{2}hL\right)^n\left[1-\left(\frac{1}{2}hL\right)^{m-n}\right]}{1-\frac{1}{2}hL}\left|y_{i+1}^{(1)}-y_{i+1}^{(0)}\right|\end{aligned} \tag{10.2-9}$$

选择 h 足够小，并且满足 $\frac{1}{2}hL<1$ 时，式（10.2-9）表明由迭代公式（10.2-6）和式（10.2-7）

所得序列式（10.2-8）为 Cauchy 序列，从而它收敛于方程（10.2-4）的解 y_{i+1}.

式（10.2-6）和式（10.2-7）是改进 Euler 法实算公式．最后一章会给出程序和计算结果，这里不再给出算例.

10.2.3 预估-校正法

预估-校正法是介于显式 Euler 法和隐式改进 Euler 法之间，为避免式（10.2-7）所采用的多次迭代而使用的一种单步迭代法，即

$$y_{i+1}^{(0)} = y_i + hf(t_i, y_i) \tag{10.2-10}$$

$$y_{i+1} = y_i + \frac{h}{2}(f(t_i, y_i) + f(t_{i+1}, y_{i+1}^{(0)})) \tag{10.2-11}$$

其中，式（10.2-10）是用 Euler 法求出 $y_{i+1}^{(0)}$ 作为 y_{i+1} 的预估值，式（10.2-11）是用改进 Euler 法由 $y_{i+1}^{(0)}$ 经过一次迭代而得到的 y_{i+1} 的更加准确的校正值．式（10.2-10）中的 $y_{i+1}^{(0)}$ 值称为预估值，式（10.2-11）则是校正，所以称这种方法为预估-校正法.

可以证明预估-校正法的局部截断误差为 $O(h^3)$，而整体截断误差为 $O(h^2)$，也就是说，预估-校正法的精度和改进 Euler 法的精度一样，但计算方便得多，因此预估-校正法是实用算法.

10.2.4 预估-校正法的计算步骤

下面介绍预估-校正法的计算步骤.

（1）计算步长 h 和各分点时间 t_i.

计算公式为

$$h = \frac{T}{n},\quad t_i = ih,\quad i = 0,1,\cdots,n.$$

（2）按式（10.2-10）和式（10.2-11）计算各时刻的 y_i 值.

10.2.5 预估-校正法的计算实例

【例 1】 用预估-校正法求初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = ty, & 0 \leqslant t \leqslant 1 \\ y(0) = 1 \end{cases}$$

的近似值.

【解】 取 $n = 10$，$h = 0.1$，按以下公式计算：

$$\begin{cases} y(0) = 1 \\ k_{i+1}^{(1)} = y_i + 0.1t_i y_i \\ k_{i+1}^{(2)} = y_i + 0.1t_{i+1}k_{i+1}^{(1)} \\ y_{i+1} = \dfrac{1}{2}(k_{i+1}^{(1)} + k_{i+1}^{(2)}) \end{cases} \qquad i = 0,1,\cdots,9$$

计算结果如表 10.2 所示.

表 10.2　预估-校正法的计算结果

i	t_i	$k_{i+1}^{(1)}$	$k_{i+1}^{(2)}$	y_i	$y(t_i)$	$\lvert y_i - y(t_i)\rvert$
0	0.0			1.000000000	1.000000000	0.000000000
1	0.1	1.000000000	1.010000000	1.005000000	1.005012521	0.000012521
2	0.2	1.015050000	1.025301000	1.020175500	1.020201340	0.000025840
3	0.3	1.040579010	1.051392870	1.045985940	1.046027860	0.000041920
4	0.4	1.077365518	1.089080561	1.083223040	1.083287068	0.000064028
5	0.5	1.126551961	1.139550638	1.133051299	1.133148453	0.000097154
6	0.6	1.189703864	1.204433531	1.197068698	1.197217363	0.000148665
7	0.7	1.268892820	1.285891195	1.277392007	1.277621313	0.000229306
8	0.8	1.366809448	1.386736763	1.376773106	1.377127764	0.000354659
9	0.9	1.486914954	1.510595452	1.498755203	1.499302500	0.000547297
10	1.0	1.633643171	1.662119520	1.647881346	1.648721271	0.000839925

由表 10.2 可知，预估-校正法所得近似值比 Euler 法所得近似值更接近理论解.

10.3　Runge-Kutta 法

Euler法和改进Euler法的局部截断误差和整体截断误差都较大，可以利用数学工具Taylor级数提高局部截断误差和整体截断误差的阶.

10.3.1　高阶 Taylor 法

Euler 法利用 1 阶 Taylor 展开式来逼近微分方程的解，这种方法的局部截断误差为 $O(h^2)$，精度仅为 1，可以通过适当提高 n 值，来改善差分法收敛的性质并提高精度.

假定初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = f(t,y), & 0 \leqslant t \leqslant T \\ y(0) = y_0 \end{cases}$$

的解 $y(t)$ 及 $f(t,y)$ 足够光滑.

将 $y(t_{i+1})$ 在 t_i 处展开成 n 阶 Taylor 级数

$$y(t_{i+1}) = y(t_i) + y'(t_i)h + \frac{y''(t_i)}{2!}h^2 + \cdots + \frac{y^{(n)}(t_i)}{n!}h^n + \frac{y^{(n+1)}(\xi_i)}{(n+1)!}h^{n+1},$$

其中，$t_i < \xi_i < t_{i+1}$，再将 $y'(t_i) = f(t_i, y(t_i))$ 代入上式，得

$$y(t_{i+1}) = y(t_i) + f(t_i, y(t_i))h + f'(t_i, y(t_i))\frac{h^2}{2} + \cdots + f^{(n-1)}(t_i, y(t_i))\frac{h^n}{n!} + f^{(n)}(\xi_i, y(\xi_i))\frac{h^{n+1}}{(n+1)!}.$$

当 h 充分小时，略去高阶无穷小量 $O(h^{n+1})$，用 y_i 和 y_{i+1} 分别代替 $y(t_i)$ 和 $y(t_{i+1})$，得

$$\begin{cases} y_0 = y(0) \\ y_{i+1} = y_i + hf(t_i, y_i) + \dfrac{h^2}{2!} f'(t_i, y_i) + \cdots + \dfrac{h^n}{n!} f^{(n-1)}(t_i, y_i) \end{cases} \tag{10.3-1}$$

称上式为 n 阶 Taylor 法.

易知，Euler 法为一阶 Taylor 法.

10.3.2 二阶 Taylor 法计算实例

【例 1】 用二阶 Taylor 法求初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = ty, & 0 \leqslant t \leqslant 1 \\ y(0) = 1 \end{cases}$$

的近似解.

【解】 首先求函数 $f(t, y(t)) = ty$ 的导数：

$$f'(t, y(t)) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} = y(1 + t^2)$$

取 $n = 10$，$h = 0.1$，则

$$\begin{cases} y_0 = 1 \\ y_{i+1} = y_i + 0.1 t_i y_i + \dfrac{(0.1)^2}{2} y_i (1 + t_i^2) \end{cases}$$

计算结果如表 10.3 所示.

表 10.3 二阶 Taylor 法的计算结果

i	t_i	y_i	$y(t_i)$	$\lvert y_i - y(t_i)\rvert$
0	0.0	1.000000000	1.000000000	0.000000000
1	0.1	1.005000000	1.005012521	0.000012521
2	0.2	1.020125250	1.020201340	0.000076090
3	0.3	1.045832406	1.046027860	0.000195454
4	0.4	1.082907165	1.083287068	0.000379903
5	0.5	1.132504313	1.133148453	0.000644140
6	0.6	1.196207681	1.197217363	0.001009682
7	0.7	1.276114354	1.277621313	0.001506959
8	0.8	1.374949411	1.377127764	0.002178354
9	0.9	1.496219949	1.499302500	0.003082551
10	1.0	1.644420535	1.648721271	0.004300736

比较二阶 Taylor 法和 Euler 法的计算结果，二阶 Taylor 法的精度提高了一阶，但从计算

角度来看，必须计算函数 $f(t,y)$ 在节点处的导数值，对许多问题而言，这可能是相当复杂的方法．因此，Taylor 级数法很少被用来解决实际问题，这里略去该算法的计算步骤．

10.3.3　二阶 Runge-Kutta 法

Runge-Kutta 法保留了高阶 Taylor 法所具有的高阶局部截断误差，同时避免了计算函数 $f(t,y)$ 的导数．

给定初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}t}=f(t,y), & 0\leqslant t\leqslant T\\ y(0)=y_0\end{cases} \tag{10.3-2}$$

确定常数 c_1，c_2 及 β 值，使数值解法

$$\begin{cases}y_0=y(0)\\ y_{i+1}=y_i+h(c_1k_1+c_2k_2)\\ k_1=f(t_i,y_i)\\ k_2=f(t_i+\beta h,y_i+\beta hk_1)\end{cases} \tag{10.3-3}$$

的局部截断误差为 $y(t_{i+1})-y(t_i)=O(h^3)$．

首先将 $y(t_{i+1})$ 在 t_i 处展开成 Taylor 级数

$$y(t_{i+1})=y(t_i)+hy'(t_i)+\frac{h^2}{2}y''(t_i)+O(h^3),$$

将

$$y'(t)=f(t,y(t))=f$$
$$y''(t)=f_t(t,y(t))+f_y(t,y(t))f(t,y(t))=f_t+f_yf$$

代入上式，得

$$y(t_{i+1})=y(t_i)+hf+\frac{h^2}{2}(f_t+ff_y)+O(h^3) \tag{10.3-4}$$

其中，f,f_t,f_y 分别表示相应函数在点 $(t_i,y(t_i))$ 处的函数值及相关偏导数值．

由式（10.3-3）得

$$y_{i+1}=y_i+h(c_1f(t_i,y_i)+c_2f(t_i+\beta h,y_i+\beta hf(t_i,y_i)))\,.$$

利用二元函数的 Taylor 展开式，将函数 $f(t_i+\beta h,y_i+\beta hk_1))$ 在 (t_i,y_i) 处展开成

$$f(t_i+\beta h,y_i+\beta hk_1)=f(t_i,y_i)+\beta hf_t(t_i,y_i)+\beta hf(t_i,y_i)f_y(t_i,y_i)+O(h^2),$$

代入上式得

$$y_{i+1}=y_i+h(c_1f+c_2f)+c_2\beta h(f_t+ff_y)+O(h^2) \tag{10.3-5}$$

在局部截断误差的假设前提 $y_i=y(t_i)$ 下，由式（10.3-4）减去式（10.3-5）得

$$y(t_{i+1})-y_{i+1}=h(c_1+c_2-1)f+h^2\left(\frac{1}{2}-c_2\beta\right)(f_t+ff_y)+O(h^3) \tag{10.3-6}$$

要使局部截断误差 $y(t_{i+1})-y_{i+1}=O(h^3)$，当且仅当

$$c_1+c_2-1=0,\quad \frac{1}{2}-c_2\beta=0.$$

即常数 c_1，c_2 及 β 满足条件

$$c_1+c_2=1,\quad c_2\beta=\frac{1}{2}. \tag{10.3-7}$$

式（10.3-7）中有三个未知数，但却只有两个方程，因此可以得到无数个局部截断误差为 $O(h^3)$ 的计算公式.

如果取 $c_1=c_2=\frac{1}{2}$，$\beta=1$，式（10.3-3）即为预估-校正公式

$$\begin{cases} y_0=y(0) \\ k_1=f(t_i,y_i) \\ k_2=f(t_i+h,y_i+hk_1) \\ y_{i+1}=y_i+\dfrac{h}{2}(k_1+k_2) \end{cases} \tag{10.3-8}$$

取 $c_1=\frac{1}{3}$，$c_1=\frac{2}{3}$，$\beta=\frac{3}{4}$，则递推公式为

$$\begin{cases} y_0=y(0) \\ k_1=f(t_i,y_i) \\ k_2=f\left(t_i+\dfrac{3}{4}h,y_i+\dfrac{3}{4}hk_1\right) \\ y_{i+1}=y_i+h\left(\dfrac{1}{3}k_1+\dfrac{2}{3}k_2\right) \end{cases} \tag{10.3-9}$$

取 $c_1=0$，$c_1=1$，$\beta=\frac{1}{2}$，则递推公式为

$$\begin{cases} y_0=y(0) \\ k_1=f(t_i,y_i) \\ k_2=f\left(t_i+\dfrac{1}{2}h,y_i+\dfrac{1}{2}hk_1\right) \\ y_{i+1}=y_i+hk_2 \end{cases} \tag{10.3-10}$$

式（10.3-8）、式（10.3-9）、式（10.3-10）是常见的二阶 Runge-Kutta 公式，局部截断误差均为 $O(h^3)$.

用形如式（10.3-3）而常数满足条件（10.3-7）的计算公式作为初值问题的近似值，局部截断误差的阶为 $O(h^3)$，其整体截断误差为 $O(h^2)$. 这种方法统称为二阶 Runge-Kutta 算法.

10.3.4　三阶和四阶 Runge-Kutta 法的计算公式

仿照二阶 Runge-Kutta 法的推导，可以导出三阶和四阶 Runge-Kutta 法．下面仅将常用的两个公式列举出来．

三阶 Runge-Kutta 法计算公式

$$\begin{cases} y_0 = y(0) \\ k_1 = f(t_i, y_i) \\ k_2 = f\left(t_i + \dfrac{1}{2}h, y_i + \dfrac{1}{2}hk_1\right) \\ k_3 = f(t_i + h, y_i - hk_1 + 2hk_2) \\ y_{i+1} = y_i + \dfrac{1}{6}h(k_1 + 4k_2 + k_3) \qquad i = 1, 2, \cdots, n-1 \end{cases} \tag{10.3-11}$$

局部截断误差的阶为 $O(h^4)$．

四阶 Runge-Kutta 法计算公式

$$\begin{cases} y_0 = y(0) \\ k_1 = f(t_i, y_i) \\ k_2 = f\left(t_i + \dfrac{1}{2}h, y_i + \dfrac{1}{2}hk_1\right) \\ k_3 = f\left(t_i + \dfrac{1}{2}h, y_i + \dfrac{1}{2}hk_2\right) \\ k_4 = f(t_i + h, y_i + hk_3) \\ y_{i+1} = y_i + \dfrac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \qquad i = 1, 2, \cdots, n-1 \end{cases} \tag{10.3-12}$$

局部截断误差的阶为 $O(h^5)$．

在实际应用中，常常采用四阶 Runge-Kutta 法．

10.3.5　四阶 Runge-Kutta 法的计算步骤

四阶 Runge-Kutta 法的计算步骤如下．

（1）计算步长 h 和各时刻 t_i．

计算公式为

$$h = \frac{T}{n}，\quad t_i = ih，\quad i = 0, 1, \cdots, n．$$

（2）按式（10.3-12）计算 y_i 值．

10.3.6 四阶 Runge-Kutta 法的计算实例

【例 2】用四阶 Runge-Kutta 法求解初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = ty, & 0 \leqslant t \leqslant 1 \\ y(0) = 1 \end{cases}$$

的近似解.

【解】取 $n=10$，$h=0.1$，利用递推关系式

$$\begin{cases} y_0 = 1 \\ k_1 = t_i y_i \\ k_2 = (t_i + 0.05)(y_i + 0.05k_1) \\ k_3 = (t_i + 0.05)(y_i + 0.05k_2) \\ k_4 = (t_i + 0.1)(y_i + 0.1k_3) \\ y_{i+1} = y_i + \dfrac{0.1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$

计算结果如表 10.4 所示. 其中理论值 $y(t_i)$ 见前几个表格.

表 10.4 四阶 Runge-Kutta 法的计算结果

i	t_i	k_1	k_2	k_3	k_4	y_i	$\lvert y_i - y(t_i)\rvert$
0	0.0	0.000000000	0.000000000	0.000000000	0.000000000	1.000000000	0.000000000
1	0.1	0.000000000	0.050000000	0.050125000	0.100501250	1.005012521	0.000000000
2	0.2	0.100501252	0.151505638	0.151888170	0.204040268	1.020201340	0.000000000
3	0.3	0.204040268	0.257600838	0.258270345	0.313808512	1.046027859	0.000000001
4	0.4	0.313808358	0.371601397	0.372612775	0.433315655	1.083287065	0.000000003
5	0.5	0.433314826	0.497228763	0.498666826	0.566576874	1.133148446	0.000000007
6	0.6	0.566574223	0.638812436	0.640798987	0.718337007	1.197217347	0.000000016
7	0.7	0.718330408	0.801537014	0.804241229	0.894349029	1.277621279	0.000000034
8	0.8	0.894334896	0.991753518	0.995406717	1.101729561	1.377127695	0.000000069
9	0.9	1.101702156	1.217380882	1.222297228	1.349421676	1.499302362	0.000000138
10	1.0	1.349372126	1.488432420	1.495037784	1.648806141	1.648721007	0.000000264

将计算结果与预估-校正法进行比较可以看出，算法大大提高了计算精度，但四阶 Runge-Kutta 法的计算量较大.

10.4 Adams 法

Euler 法、预估-校正法和 Runge-Kutta 法，在计算 y_{i+1} 时只用到前一个节点上的近似值 y_i，这类方法统称为单步法. 实际上，经过多次单步法计算得到一系列近似值 $y_1, y_2, \cdots, y_i$ 后，

如果利用这些已经求出的近似值来计算 y_{i+1}，也可获得较高的精度.

【定义】给定初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}t}=f(t,y), & 0\leqslant t\leqslant T\\ y(0)=y_0\end{cases} \tag{10.4-1}$$

采用以下计算公式求 y_{i+1}：

$$y_{i+1}=a_1y_i+a_2y_{i-1}+\cdots+a_ky_{i-k+1}+h(b_0f_{i+1}+b_1f_i+\cdots+b_kf_{i-k+1}) \tag{10.4-2}$$

上式简写为

$$y_{i+1}=\sum_{j=1}^{k}a_jy_{i+1-j}+h\sum_{j=0}^{k}b_jf_{i+1-j},$$

其中，a_j，b_j 为常数，$f_{i+1-j}=f(t_{i+1-j},y_{i+1-j})$，$k\geqslant 1$ 为整数.

式(10.4-2)是用前若干节点处的函数值和导数值的线性组合来计算 $y(t_{i+1})$ 的近似值 y_{i+1}，这种方法统称为线性多步法. 当 $k=1$ 时称为单步法，$k>1$ 时称为多步法或称为 k 步法. 当 $b_0=0$ 时称为显式算法. $b_0\neq 0$ 时称为隐式算法.

式（10.4-2）中的系数可由幂级数展开的待定系数法确定，也可由数值积分导出.

将常微分方程（10.4-1）两边从 t_i 到 t_{i+1} 积分，写成等价积分方程

$$y(t_{i+1})=y(t_i)+\int_{t_i}^{t_{i+1}}f(t,y(t))\mathrm{d}t \tag{10.4-3}$$

由于式（10.4-3）中的函数 f 包含未知函数 $y(t)$，因此无法直接算出右边的积分. 但是，可利用数值积分公式导出一系列的计算公式.

10.4.1　Adams 内插法

对式（10.4-3）中的 $f(t,y)$ 用代数插值多项式逼近时，若节点含 t_{i+1}，那么这类 Adams 多步算法称为 Adams 内插法.

通常，教材会给出 Newton 插值多项式，然后将插值多项式在 $[t_i,t_{i+1}]$ 上积分，得出计算公式，由于 $f(t_{i+1},y_{i+1})$ 本身是未知的，用 Newton 插值多项式积分并算出系数的理论值不易，所以一般教材不详细介绍积分公式的推导过程，仅给出计算公式. 这里用幂级数型插值多项式逼近 $f(t,y)$ 作为被积函数在 $[t_i,t_{i+1}]$ 上的积分，并给出二步内插公式和三步内插公式的整个推导过程.

1．二步 Adams 内插公式推导及截断误差

取 t_{i+1}，t_i，t_{i-1} 为节点，设

$$\begin{cases} Z_2(t)=\sum_{j=0}^{2}a_j(t-t_i)^j \\ Z_2(t_{i-1})=f(t_{i-1},y_{i-1})=f_{i-1} \\ Z_2(t_i)=f(t_i,y_i)=f_i \\ Z_2(t_{i+1})=f(t_{i+1},y_{i+1})=f_{i+1} \end{cases},$$

即

$$\begin{cases} a_0=f_i \\ a_0+a_1h+a_2h^2=f_{i+1} \\ a_0-a_1h+a_2h^2=f_{i-1} \end{cases},$$

解之得

$$\begin{cases} a_0=f_i \\ a_1=\dfrac{f_{i+1}-f_{i-1}}{2h} \\ a_2=\dfrac{f_{i+1}-2f_i+f_{i-1}}{2h} \end{cases},$$

$$\int_{t_i}^{t_{i+1}}Z_2(t)\mathrm{d}t=h\left(\frac{5}{12}f_{i+1}+\frac{2}{3}f_i-\frac{1}{12}f_{i-1}\right).$$

由此可得出二步 Adams 内插公式为

$$y_{i+1}=y_i+h\left(\frac{5}{12}f_{i+1}+\frac{2}{3}f_i-\frac{1}{12}f_{i-1}\right) \tag{10.4-4}$$

二步 Adams 内插公式（10.4-4）是一隐式算式，和改进 Euler 法一样，必须用迭代法才能得出解．二步 Adams 内插公式的实际算式为

$$\begin{cases} y_{i+1}^{(m+1)}=y_i+h\left(\dfrac{5}{12}f(t_{i+1},y_{i+1}^{(m)})+\dfrac{2}{3}f(t_i,y_i)-\dfrac{1}{12}f(t_{i-1},y_{i-1})\right) & i=2,3,\cdots,n-1 \\ y_{i+1}^{(0)}=y_i+hf(t_i,y_i) & \\ \left|y_{i+1}^{(m+1)}-y_{i+1}^{(m)}\right|>\varepsilon & m=0,1,\cdots \end{cases}$$

上式中 ε 为迭代精度，和改进 Euler 法一样取 Euler 法计算结果为迭代过程初值．

对于常微分方程初值问题，只有 $y_0=y(0)$ 是给出的，上式无法算出 y_1，y_1 必须用精度一致或精度较高的显式算法给出，例如用四阶 Runge-Kutta 法给出．

$Z_2(t)$ 是二次多项式，但是 $\int_{t_i}^{t_{i+1}}Z_2(t)\mathrm{d}t$ 不能直接用 Simpson 积分估计误差，因为所取节点不是 t_i，$\dfrac{t_i+t_{i+1}}{2}$，t_{i+1}，所以只能用插值余项来估计积分的截断误差：

$$\begin{aligned}\varepsilon_{i+1} &= \int_{t_i}^{t_{i+1}} \big(f(t,y)-Z_2(t)\big)\mathrm{d}t \\ &= \int_{t_i}^{t_{i+1}} \frac{f'''(\zeta)}{3!}(t-t_{i+1})(t-t_i)(t-t_{i-1})\mathrm{d}t \\ &= \frac{f'''(\xi)}{3!}\int_0^h (t-h)t(t+h)\mathrm{d}t \\ &= \frac{f'''(\xi)}{3!}\int_0^h (t^2-h^2)t\mathrm{d}t \\ &= -\frac{1}{24}h^4 f'''(\xi)\end{aligned} \tag{10.4-5}$$

上式即为二步 Adams 算法的截断误差计算公式.

2．三步 Adams 内插法公式及截断误差

用同样的方法可给出三步 Adams 内插计算公式及截断误差：

$$\begin{cases} y_{i+1} = y_i + \dfrac{h}{24}(9f_{i+1}+19f_i-5f_{i-1}+f_{i-2}) \\ \varepsilon_{i+1} = -\dfrac{19}{720}h^5 f^{(5)}(\xi) \end{cases} \tag{10.4-6}$$

式（10.4-6）仍然是隐式格式，也须用迭代法求解，其实际计算公式为

$$\begin{cases} y_{i+1}^{(m+1)} = y_i + \dfrac{h}{24}(9f(t_{i+1},y_{i+1}^{(m)})+19f_i-5f_{i-1}+f_{i-2}) \\ y_{i+1}^{(0)} = y_i + hf(t_i,y_i) \\ \left|y_{i+1}^{(m+1)} - y_{i+1}^{(m)}\right| > \varepsilon \end{cases}, \quad \begin{matrix} i=3,4,\cdots,n-1 \\ m=0,1,\cdots \end{matrix}$$

上式中 y_1，y_2 可用四阶 Runge-Kutta 法算出.

还可给出更高阶的 Adams 内插公式.

由于上述公式的左右两端都含有 y_{i+1}，且右端的 y_{i+1} 含在二元函数 $f(t,y)$ 中，故上述算法称为 Adams 隐式算法.

式（10.4-5）中 $\xi\in[t_{i-1},t_{i+1}]$，ξ 可能在积分区域之外，无论是 Adams 外推法还是内插法都存在这一问题.

10.4.2　Adams 外插法

当 t_{i+1} 在用以代替函数 $f(t,y)$ 的插值多项式所取的节点之外时，其算法称为 Adams 外插法.

1．三步 Adams 外插法公式推导及截断误差

取 t_i，t_{i-1}，t_{i-2} 为节点，设

$$\begin{cases} Z_2(x)=\sum_{j=0}^{2} a_j (x-t_{i-1})^j \\ Z_2(t_i)=f(t_i,y_i)=f_i \\ Z_2(t_{i-1})=f(t_{i-1},y_{i-1})=f_{i-1} \\ Z_2(t_{i-2})=f(t_{i-2},y_{i-2})=f_{i-2} \end{cases},$$

易算出

$$\begin{cases} a_0=f_{i-1} \\ a_1=\dfrac{f_i-f_{i-2}}{2} \\ a_2=\dfrac{f_i-2f_{i-1}+f_{i-2}}{2} \end{cases},$$

$$\int_{t_i}^{t_{i+1}} Z_2(t)\mathrm{d}t = h\left(\frac{23}{12}f_i-\frac{4}{3}f_{i-1}+\frac{5}{12}f_{i-2}\right).$$

由此可得出三步 Adams 外插法的计算公式为

$$y_{i+1}=y_i+h\left(\frac{23}{12}f_i+\frac{4}{3}f_{i-1}+\frac{5}{12}f_{i-2}\right),$$

$$\begin{aligned} \varepsilon_{i+1} &= \int_{t_i}^{t_{i+1}} \frac{f^{(3)}(\zeta)}{3!}(t-t_i)(t-t_{i-1})(t-t_{i-2})\mathrm{d}t \\ &= O(h^4). \end{aligned}$$

请同学们自己推导具体结果.

上式即为三步 Adams 外插算法的截断误差计算公式.

2．四步 Adams 外插法公式及截断误差

这里直接给出四步 Adams 外插计算公式及截断误差：

$$\begin{cases} y_{i+1}=y_i+\dfrac{h}{24}\left(55f_i-59f_{i-1}+37f_{i-2}-9f_{i-3}\right) \\ \varepsilon_{i+1}=\dfrac{251}{720}h^5 f^{(5)}(\xi) \end{cases} \tag{10.4-8}$$

由于可利用式（10.4-7）和式（10.4-8）直接得 y_{i+1}，故 Adams 外插法是显式算法.

注意：由于 $f(t_{i+1},y_{i+1})$ 是未知的，所以不能用第 6 章中的公式递推计算各系数，而是用 Newton-Leibniz 积分公式计算结果．同样，计算 Newton 插值多项式系数时也不能用第 6 章中的公式递推计算得出结果．这正是一般教材未给出 Adams 各计算公式的推导过程的原因.

10.4.3　Adams 外插法与内插法的计算实例

【例 1】用 Adams 四步外插法、二步和三步内插法求解初值问题

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}t}=ty, & 0\leqslant t\leqslant 1\\ y(0)=1\end{cases}$$

的近似解.

【解】取 $n=10$，$h=0.1$，其理论解为 $y=\mathrm{e}^{t^2/2}$，表 10.5 中空白部分的 y_i 取理论值，一般可采用四阶 Runge-Kutta 法来计算.

表 10.5　Adams 四步显式法、三步隐式法和二步隐式法的计算结果

i	t_i	四步显式法 y_i	$\lvert y_i-y(t_i)\rvert$	三步隐式法 y_i	$\lvert y_i-y(t_i)\rvert$	二步隐式法 y_i	$\lvert y_i-y(t_i)\rvert$
0	0.0						
1	0.1						
2	0.2					1.020214388	0.000013048
3	0.3			1.046028565	0.000000705	1.046055418	0.000027558
4	0.4	1.083276318	0.000010750	1.083288995	0.000001927	1.083331697	0.000044629
5	0.5	1.133119504	0.000028949	1.133152239	0.000003786	1.133214028	0.000065575
6	0.6	1.197161221	0.000056143	1.197223844	0.000006481	1.197309457	0.000092094
7	0.7	1.277525842	0.000095471	1.277631615	0.000010302	1.277747747	0.000126434
8	0.8	1.376976483	0.000151281	1.377143431	0.000015667	1.377299396	0.000171631
9	0.9	1.499072688	0.000229812	1.499325673	0.000023173	1.499534350	0.000231850
10	1.0	1.648381258	0.000340012	1.648754945	0.000033675	1.649034170	0.000312900

比较 Adams 显式法和隐式法：

（1）隐式法系数的绝对值比较小，因此用隐式法计算 f_i 时舍入误差较小；

（2）显式法的计算量比隐式法的计算量小；

（3）从有关文献中可知，显式法的绝对稳定域比隐式法的绝对稳定域小.

10.4.4　四阶 Adams 预估-校正法的计算公式

当函数 $f(t,y)$ 为非线性函数时，用 Adams 隐式法无法直接解出 y_{i+1}，而需要进行迭代，这样会增加工作量，因此在实际中往往把显式法和隐式法结合使用，从而导出下列预估-校正法．以四阶 Adams 法为例，先由显式法算出近似值，作为隐式法的预测值，再进行校正．具体步骤如下：

（1）用四阶 Runge-Kutta 法，由初值 y_0 计算出 y_1,y_2,y_3；

（2）用四阶显式法预估

$$y_i^{(0)}=y_{i-1}+\frac{h}{24}(55f_{i-1}-59f_{i-2}+37f_{i-3}-9f_{i-4}),\quad i=4,5,\cdots,n \tag{10.4-9}$$

（3）将用显式法计算出的 $y_i^{(0)}$ 代入隐式法，得

$$y_i = y_{i-1} + \frac{h}{24}(9f_i^{(0)} + 19f_{i-1} - 5f_{i-2} + f_{i-3})，\quad i = 4,5,\cdots,n \tag{10.4-10}$$

其中，$f_i^{(0)} = f(t_{i+1}, y_i^{(0)})$.

10.4.5 四阶 Adams 预估-校正法的计算步骤

四阶预估-校正法的计算步骤如下.

（1）计算步长 h 和各时刻 t_i.

计算公式为

$$h = \frac{T}{n}，\quad t_i = ih，\quad i = 0,1,\cdots,n.$$

（2）用四阶 Runge-Kutta 法计算 y_1, y_2, y_3 值；

（3）用 Adams 预估-校正法求 $y_4, y_5, \cdots, y_n$ 值.

计算公式为

$$\begin{cases} y_i^{(0)} = y_{i-1} + \dfrac{h}{24}(55f_{i-1} - 59f_{i-2} + 37f_{i-3} - 9f_{i-4}) \\ f_i^{(0)} = f(t_i, y_i^{(0)}) \\ y_i = y_{i-1+}\dfrac{h}{24}(9f_i^{(0)} + 19f_{i-1} - 5f_{i-2} + f_{i-3}) \end{cases}，\quad i = 4,5,\cdots,n$$

四阶 Adams 预估-校正法的截断误差和四阶 Adams 内插法的截断误差在同一量级.

10.4.6 四阶 Adams 预估-校正法的计算实例

【例 2】 用四阶 Adams 预估-校正法求解初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = ty, & 0 \leqslant t \leqslant 1 \\ y(0) = 1 \end{cases}$$

的近似解.

【解】 取 $n = 10$，$h = 0.1$，计算结果如表 10.6 所示.

表 10.6 四阶 Adams 预估-校正法的计算结果

i	t_i	$y_i^{(0)}$	y_i	$y(t_i)$	$\lvert y_i - y(t_i)\rvert$
0	0.0		1.000000000	1.000000000	
1	0.1		1.005012521	1.005012521	
2	0.2		1.020201340	1.020201340	
3	0.3		1.046027859	1.046027860	
4	0.4	1.083276318	1.083288082	1.083287068	0.000001014

（续表）

i	t_i	$y_i^{(0)}$	y_i	$y(t_i)$	$\lvert y_i - y(t_i)\rvert$
5	0.5	1.133132346	1.133150929	1.133148453	0.000002476
6	0.6	1.197195090	1.197221844	1.197217363	0.000004481
7	0.7	1.277591664	1.277628487	1.277621313	0.000007174
8	0.8	1.377088898	1.377138524	1.377127764	0.000010760
9	0.9	1.499251792	1.499318018	1.499302500	0.000015518
10	1.0	1.648655032	1.648743098	1.648721271	0.000021828

注意：表中的 y_1, y_2, y_3 值直接取自前面的四阶 Runge-Kutta 法的计算结果.

10.5　收敛性与稳定性

收敛性与稳定性从两个不同角度描述了微分方程数值解法的实用价值，只有既收敛又稳定的方法，才可能提供比较可靠的计算结果.

10.5.1　收敛性

收敛性问题讨论的是在 $h \to 0$ 时数值解 y_i 能否收敛到方程理论解 $y(t_i)$ 的问题.

本节仅讨论单步法的收敛性.

【定义 1】设 $y(t_i)$ 表示微分方程的理论解，y_i 表示从差分方程所求得的第 i 步的近似解，如果满足 $\lim\limits_{h\to 0}\max\limits_{1\leqslant i\leqslant n}\left|y(t_i)-y_i\right|=0$，则称此数值方法对原微分方程是收敛的.

Euler 法、Taylor 法、Runge-Kutta 法是单步法，它们都可以用下面的一般形式表示：

$$y_{i+1} = y_i + hp(t_i, y_i, h) \tag{10.5-1}$$

例如，对 Euler 法，可取

$$p(t_i, y_i, h) = f(t_i, y_i).$$

对四阶 Runge-Kutta 法，可取

$$p(t_i, y_i, h) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

其中 k_1，k_2，k_3，k_4 如 10.3 节的算法中所示.

对于式（10.5-1），局部截断误差的一般表示式为

$$\varepsilon_i = y(t_{i+1}) - y_{i+1} = y(t_{i+1}) - y_i - hp(t_i, y_i, h),$$

由于局部截断误差的前提是 $y_i = y(t_i)$，所以

$$\varepsilon_i = y(t_{i+1}) - y(t_i) - hp(t_i, y(t_i), h) \tag{10.5-2}$$

从定义 1 中可看出，数值方法的收敛性并不涉及计算过程中的舍入误差，而只与方法的整体截断误差有关.

【定理 1】假设对一初值问题

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = f(t,y), & a \leqslant t \leqslant b \\ y(0) = y_0 \end{cases} \tag{10.5-3}$$

用以下形式的单步法求近似解：

$$\begin{cases} y_0 = y(0) \\ y_{i+1} = y_i + hp(t_i, y_i, h) & i = 0,1,\cdots,n \end{cases} \tag{10.5-4}$$

其中，函数 $p(t,y,h)$ 满足 $p(t,y,0) = f(t,y)$，如果存在整数 $h_0 > 0$，使得 $p(t,y,0)$ 在区间

$$D = \{(t,y,h) \mid 0 \leqslant t \leqslant T,\ -\infty \leqslant y \leqslant +\infty, 0 \leqslant h \leqslant h_0\}$$

上连续，且关于 y 满足 Lipschitz 条件，即存在与 t，h 无关的正常数 L，满足以下不等式：

$$\left|p(t,y_1,h) - p(t,y_2,h)\right| \leqslant L\left|y_1 - y_2\right| \tag{10.5-5}$$

再设递推公式（10.5-4）所得数值解的局部截断误差满足

$$\left|\varepsilon_i\right| \leqslant Ch^{q+1} \tag{10.5-6}$$

则对于整体截断误差 $\left|E_i\right|$ 满足估计式

$$\left|E_i\right| \leqslant \mathrm{e}^{L(b-a)}\left|E_0\right| + \frac{Ch^q}{L}(\mathrm{e}^{L(b-a)} - 1) \tag{10.5-7}$$

【证明】由式（10.5-2），得

$$y(t_{i+1}) = y(t_i) + hp(t_i, y(t_i), h) + \varepsilon_i ,$$

上式减去式（10.5-4），得

$$y(t_{i+1}) - y_{i+1} = y(t_i) - y_i + hp(t_i, y(t_i), h) - hp(t_i, y_i, h) + \varepsilon_i .$$

利用式（10.5-5），得

$$\left|y(t_{i+1}) - y_{i+1}\right| \leqslant \left|y(t_i) - y_i\right| + hL\left|y(t_i) - y_i\right| + \left|\varepsilon_i\right| .$$

令 $E_i = y(t_i) - y_i$，并利用式（10.5-6），得

$$\left|E_{i+1}\right| \leqslant \left|E_i\right| + hL\left|E_i\right| + Ch^{q+1} = (1+hL)\left|E_i\right| + Ch^{q+1} ,$$

从而

$$\begin{aligned} \left|E_i\right| &\leqslant (1+hL)\left|E_{i-1}\right| + Ch^{q+1} \\ &\leqslant (1+hL)((1+hL)\left|E_{i-2}\right| + Ch^{q+1}) + Ch^{q+1} \\ &\cdots \\ &\leqslant (1+hL)^i\left|E_0\right| + (1 + (1+hL) + \cdots + (1+hL)^{i-1}) + Ch^{q+1} \\ &= (1+hL)^i\left|E_0\right| + \frac{Ch^{q+1}}{hL}(1 + (1+hL)^i - 1). \end{aligned}$$

利用 10.1 节的引理 1，得

$$(1+hL)^i < \mathrm{e}^{ihL} ,$$

于是

$$\begin{aligned}|E_i| &\leqslant \mathrm{e}^{ihL}|E_0| + \frac{Ch^q}{L}(\mathrm{e}^{ihL}-1)\\ &\leqslant \mathrm{e}^{nhL}|E_0| + \frac{Ch^q}{L}(\mathrm{e}^{nhL}-1)\\ &\leqslant \mathrm{e}^{L(b-a)}|E_0| + \frac{Ch^q}{L}(\mathrm{e}^{L(b-a)}-1).\end{aligned}$$

定理 1 表明：

（1）如果某个形如式（10.5-1）的单步法的局部截断误差的阶为 $O(h^{q+1})$，则其整体截断误差的阶为 $O(h^q)$；

（2）在没有初始误差即 $|E_i|=0$ 时，只要式（10.5-1）中的 $p(x,y,h)$ 关于 y 满足 Lipschitz 条件，由式（10.5-5）便可得到

$$y(t_i)-y_i=|E_i|\to 0 \quad (h\to 0),$$

这表明单步法公式（10.5-4）的数值 y_i 在 $h\to 0$ 时收敛于真解 $y(t)$ 在 $t=t_i$ 处的值．因此，常将该定理称为单步法的收敛性定理．

10.5.2　稳定性

这里所说的稳定性，不是指常微分方程初值问题［即式（10.5-3）］本身的稳定性，而是指数值方法的稳定性，即数值稳定性．

即使是一个收敛的方法，在实际计算中，由于初始值一般都带有误差，同时，在计算过程中还常产生舍入误差，这些误差又必然会传播下去，对以后的计算结果产生影响，因此，还应该关心实际计算出的数值解能否作为微分方程理论解的近似结果．数值稳定性问题讨论的是这种误差的积累能否得到控制的问题．

本节只讨论绝对稳定性．

【定义 2】对于给定初值问题［即式（10.5-3）］，步长 h 固定，用某一数值方法计算 y_i 时，实际得到的计算结果为 $\tilde{y}_i$，且由误差 $\delta_i=y_i-\tilde{y}_i$ 引起的以后各节点处 $y_j(j>i)$ 的误差为 δ_j，如果 $|\delta_j|\leqslant|\delta_i|$，那么就称这个数值方法是绝对稳定的．

一个数值方法是否绝对稳定，不仅与方法本身有关，而且还与函数 $f(t,y)$ 及步长 h 有关．为了便于讨论，常用试验方程

$$\frac{\mathrm{d}y}{\mathrm{d}t}=\lambda y$$

（其中 λ 为复常数）来讨论数值方法的绝对稳定性．

设 λ 为复数，在 λh 复平面上，某一数值方法对试验方程绝对稳定的复数 λh 的全体所组成的集合，称为该数值方法的绝对稳定域．

由式（10.5-2）可知，线性多步法（包括单步法）的计算公式为

$$y_{i+1}=\sum_{j=1}^{k}a_j y_{i+1-j}+h\sum_{j=0}^{k}b_j f_{i+1-j}\ ,\quad i=k-1,k+1,\cdots$$

【定义 3】称多项式

$$P(\xi)=\xi^k-a_k\xi^{k-1}-a_{k-1}\xi^{k-2}-\cdots-a_2\xi-a_1$$

为多步法的特征多项式.

【定理 2】若数值方法的特征方程 $P(\xi)=0$ 的根为 ξ_i，并且 $|\xi_i|\leqslant 1$（$i=1,2,\cdots,k$），则此数值方法是稳定的.

证明略.

【例 1】讨论 Euler 法

$$y_{i+1}=y_i+hf(t_i,y_i)$$

的绝对稳定性.

【解】Euler 法作用于试验方程 $\dfrac{\mathrm{d}y}{\mathrm{d}t}=\lambda y$ 时，成为

$$y_{i+1}=y_i+\lambda hf_i\ ,$$

其特征方程为

$$P(\xi)=\xi-(1+\lambda h)\ ,$$

其特征根为

$$\xi=1+\lambda h\ .$$

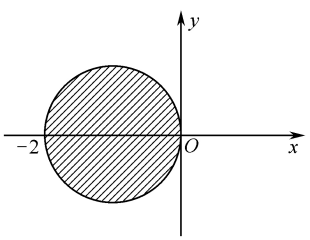

图 10.1 Euler 法绝对稳定域

由于 λh 表示复数，所以当 λh 落在以 $(-1,0)$ 为圆心、以 1 为半径的圆形区域时，$|\xi_i|\leqslant 1$. 因此 Euler 法是稳定的，其特定区域如图 10.1 所示.

【例 2】讨论 Runge-Kutta 法的绝对稳定域

【解】以四阶 Runge-Kutta 法为例，并将其作用到 $\dfrac{\mathrm{d}y}{\mathrm{d}t}=\lambda y$，有

$$y_{i+1}=y_i+\frac{h}{6}\left(k_1+2k_2+2k_3+k_4\right)$$

$$k_1=f(t_i,y_i)=\lambda y_i h$$

$$k_2=f\left(t_i+\frac{1}{2}h,y_i+\frac{1}{2}hk_1\right)=\lambda y_i+\frac{h}{2}\lambda^2 y_i$$

$$k_3=f\left(t_i+\frac{1}{2}h,y_i+\frac{1}{2}hk_2\right)=\lambda y_i+\frac{1}{2}h\lambda^2 y_i+\frac{1}{4}h^2\lambda^3 y_i$$

$$k_4=f(t_i+h,y_i+hk_3)=\lambda y_i+\lambda^2 h y_i+\frac{h}{2}\lambda^3 y_i+\frac{1}{4}\lambda^4 h^3 y_i$$

所以

$$y_{i+1}=\left(1+h\lambda+\frac{1}{2}h^2\lambda^2+\frac{1}{6}h^3\lambda^3+\frac{1}{24}\lambda^4 h^4\right)y_i\ .$$

令 $\mu = h\lambda$，则

$$y_{i+1} = \left(1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{6}\mu^3 + \frac{1}{24}\mu^4\right) y_i ,$$

其特征方程为

$$P(\xi) = \xi - \left(1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{6}\mu^3 + \frac{1}{24}\mu^4\right),$$

其特征根为

$$\xi = 1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{6}\mu^3 + \frac{1}{24}\mu^4 .$$

由此可知,绝对稳定性区域为在 $\mu = h\lambda$ 复平面上满足

$$\left|1 + \mu + \frac{1}{2}\mu^2 + \frac{1}{6}\mu^3 + \frac{1}{24}\mu^4\right| \leqslant 1$$

的全体复数，如图 10.2 所示.

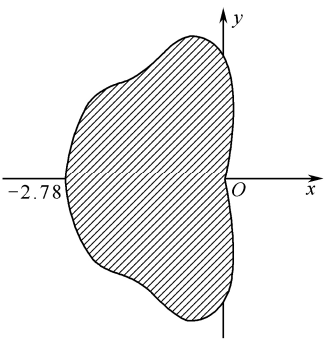

图 10.2　四阶 Runge-Kutta 法的绝对稳定域

习题 10

1．分别用 Euler 法和预估-校正法求初值问题

$$\begin{cases} y' = x^2, & x \in [0,2] \\ y(0) = 1 \end{cases}$$

的数值解（取 $h = 0.5$），并将计算结果与理论解 $y = \frac{1}{3}x^3 + 1$ 进行比较.

2．用四阶 Runge-Kutta 法求初值问题 $\begin{cases} y' = y \\ y(0) = 1 \end{cases}$ 在 $x = 0.2,\ 0.4,\ 0.6,\ 0.8,\ 1$ 处的近似值.

3．确定常数 a，b，c，d，使递推公式 $y_{i+1} = ay_{i-1} + h(by'_{i+1} + cy'_i + dy'_{i-1})$ 的局部截断误差的阶为 $O(h^4)$.

4．用四步 Adams 外推法求解下面的初值问题：

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = y, & t \in [0,1] \\ y(0) = 1 \end{cases}$$

5．用三步 Adams 内插法求解下面的初值问题，要求计算精度为 10^{-5}：

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = y, & t \in [0,1] \\ y(0) = 1 \end{cases}$$

6．用三步 Adams 预估-校正法求解下面的初值问题：

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = y, & t \in [0,1] \\ y(0) = 1 \end{cases}$$

7．证明 Euler 法的绝对稳定区域包含在四阶 Runge-Kutta 法的绝对稳定区域内.

第 11 章　算法、公式、程序和语句

在计算机盛行的今天，计算方法已不是人们手算使用的算法，而是计算机使用的算法. 计算机只能直接或间接识别人们编写的程序. 因此，计算方法不能脱离计算机，不能不考虑程序设计，程序也是算法，是计算机使用的算法，两种算法当然有联系.

不同的物理问题有时可能归纳成同一数学公式，例如扭转和受正压的薄膜都满足同一偏微分方程. 不同的计算方法都能归结成同类算法，同类算法具有相同的基本程序结构.

前面所介绍的计算方法是按数学内容分类的，为了便于设计程序，还应按程序结构分类. 一个计算方法有时由一组算式组成，有时由多组算式组成，例如 Gauss 消元法就是由消元和回代两组公式组成的. 又如 Romberg 积分，它由变步长梯形积分和外推法计算积分两部分组成，两种算法差异也很大，自然所对应的程序结构也不相同. 由一组算式组成的算法有时也并非只有一个公式.

附带指出，一组算式对应一个程序段，一个程序段所对应的程序图至少应有一个环，程序段之间的环不应有公共弧. 这是程序段的划分准则.

按程序结构划分可分为下列算法：

（1）简单算法和重复型简单算法；

（2）穷举法；

（3）递推算法；

（4）迭代算法.

所有递归算法都可改写成递推算法，故不介绍递归算法.

11.1　简单算法和重复型简单算法

计算方法的书中所介绍的算法内，有不少算法由一组公式组成，有些算法包含简单算法.

11.1.1　简单算法

简单算法是指由初等函数、变量、常数组成的可用一个算术表达式表示的算法.

简单算法包罗面最广，但都用一个赋值语句实现. 在计算机语言教材中讨论程序入门之时，所介绍的算法（计算公式）通常都属于简单算法，例如 $E=\frac{1}{2}mv^2, R=\frac{V}{I}, \cdots$；计算方法所介绍的不少算法带有简单算法，例如解常微分方程时，输入步数 n 及计时区间 $[0,T]$ 确定后，步长计算

$$h=\frac{T}{n} \tag{11.1-1}$$

就属于简单算法.

简单算法所对应的语句为赋值语句，McCabe 所给出的程序复杂程度定义如下：程序复杂程度等于程序图中线性无关环的个数，即程序中 if 的个数和循环总重数[①]，赋值语句不影响程序复杂程度，即简单算法所对应的赋值语句虽是程序的一部分，但是与程序设计中的设计关系不大.

11.1.2　重复型简单算法

重复型简单算法是指重复执行确定次数的同一简单算法，并且前一结果与当前结果无关.

这里的简单算法与前面介绍的略有不同，它所对应的语句虽然都是赋值语句，但算式不仅仅是计算，还要包括交换数据和传递数据.

重复型简单算法实际上包括下述内容.

（1）交换 Gauss 列主元素法中的两行元素.

假设已确定了 $a_{pi}^{(i)}$ 是列主元，此时必须对第 p 行元素和第 i 列元素交换行位置，计算公式为

$$a_{pi}^{(i)} \Leftrightarrow a_{ij}^{(i)},\quad j=i,i+1,\cdots,n \tag{11.1-2}$$

其中“$\Leftrightarrow$”表示左右两端数据交换.

所对应的程序语句为

```
for(j=i; j<=n; j++)
{
    v=a[p][j];
    a[p][j]=a[i][j];
    a[i][j]=v;
}
```

若用的是全主元法，则既要交换行，又要交换列.

（2）Jacobi 迭代法完成一次迭代后，必须进行数据传递.

设 $x2_i$ 是本次迭代所得结果，$x1_i$ 是上一次迭代结果，在进行下一次迭代时，必须将 $x2_i$ 的值传给 $x1_i$，计算公式为

$$x1_i^{(m+1)}=x2_i^{(m)},\quad i=1,2,\cdots,n \tag{11.1-3}$$

所对应的程序语句为

```
for(i=1; i<n; i++)
{
    X1[i]=x2[i];
}
```

在本教材中，多次使用上标，但介绍它所对应的语句中常不含上标．程序中含上标，可读性好，算法和程序对应关系更鲜明，但会大大增加内存开销．对于迭代法而言，迭代次数

① 后者由主编张世禄给出，它与原公式等价，证明略.

通常是事前不知道的，程序设计时如用数组存储，数组的大小不好确定，再加上所有迭代法总是用当前值代替上一次的值，因而通常不用数组.

（3）用迭代法 $\boldsymbol{x}^{(m+1)}=B\boldsymbol{x}^{(m)}+g\boldsymbol{x}^{(m+1)}$ 计算 $\boldsymbol{x}^{(m+1)}$ 时，要对算式进行预处理，此时也要用到重复简单算法，例如简单迭代的计算公式为

$$x_i^{(m+1)}=\frac{b_i-\sum\limits_{\substack{j=1\\ j\neq i}}^{n}a_{ij}x_j^{(m)}}{a_{ii}} \qquad (11.1\text{-}4)$$

$$x_i^{(m+1)}=b_i^*-\sum_{\substack{j=1\\ j\neq i}}^{n}a_{ij}^*x_j^{(m)}$$

$$\begin{cases} b_i^*=\dfrac{b_i}{a_{ii}}, & i=1,2,\cdots,n \\ a_{ij}^*=\dfrac{a_{ij}}{a_{ii}}, & i=1,2,\cdots,n;j\neq i \end{cases} \qquad (11.1\text{-}5)$$

式（11.1-4）和式（11.1-5）也属于重复型简单算法．用重复型简单算法计算向量 $\boldsymbol{b}$ 的基本程序模块雷同．式（11.1-5）中 a_{ij}^* 的计算也与上面的模块类似，不同的是要用到两重循环.

11.2 尝试法

11.2.1 尝试法

尝试法的定义是，在一定范围内用一个或一组条件确定的一个或多个、一组或多组解的算法.

上述范围称为尝试范围，尝试范围由循环重数及各重循环的循环条件决定，循环重数等于尝试法中涉及的变量个数，循环条件为这些变量的下标的取值范围（循环条件为这些变量的下标的取值范围或整型变量的取值范围，而尝试则是用一个或一组条件比较所选范围内的数，选择满足条件的一个数或一组数）.

尝试法中循环重数由涉及的变量个数确定，但尝试范围不是“一定”的，可大可小.

【例】有一个 4 位数，其十位和个位数相等，千位和百位数相等，该数是一个完全平方数，求该数.

【解】设该数为 x，该数的千位和百位数字为 i，十位和个位数为 j，并设对应的完全平方数为 k，由题意有 $x=iijj=k^2$，即

$$\left\{ x \middle| \begin{matrix} i=1,2,\cdots,9 \\ j=1,2,\cdots,9 \\ k=1,2,\cdots,99 \end{matrix} \quad 1100i+11j=k^2, x=1100i+11j \right\} \qquad (11.2\text{-}1)$$

式（11.2-1）对应一个三重循环，i 和 j 取值为 1～9，k 取值为 1～99，实际 k 也可取

32～99.

计算方法中所涉及的尝试法都属于穷举法．穷举中，各变量的取值范围是确定的，尝试过程是逐一尝试，不能漏掉一个，计算方法不专门讨论穷举法，穷举法仅是一个算法中的一小部分，而且仅包括下述内容．

（1）计算

$$S = \|x\|_\infty = \max_i |x_i| \tag{11.2-2}$$

（2）求列主元行位置 P，

$$P = \{j \mid j = i, i+1, \cdots, n;\ |a_{pi}| = \max_j |a_{ji}|\} \tag{11.2-3}$$

式（11.2-2）所对应的程序语句为

```
s=fabs(x[1];
for(i=2; i<=n; i++)
{
   if(fabs(x[i])>s)
      s=fabs(x[i]);
}
```

式（11.2-3）所对应的程序语句为

```
p=i;
for(j=i+1; j<=n; j++)
{
   if(fabs(a[j][i])> fabs(a[p][i]))
      p=j;
}
```

Gauss 全主元素消元法还要找行主元素．找行主元素的程序结构和找列主元的相同．求矩阵的行范数、列范数也要用到穷举法．

11.2.2　不定重循环问题

用尝试法编写程序时，若用以表示尝试范围的循环重数本身是一个变量或者循环重数很多时，常用三重循环加上转移语句模拟多重循环，这类问题实质上是不定重循环问题．这类算法在不少教材和文献中称为回溯法．这种提法欠妥，回是沿来路反向行之，溯是逆水而行，因教材是讲计算方法而不专讲程序设计，故这里只介绍一个实例，让读者清楚问题是不定重循环问题，使用的算法和回溯不挂钩．

【例 1】有 $M \times N$ 面彩旗，彩旗颜色共 N 种，每种颜色各 M 面，请编一排列程序．

【解】就本例而言，对于同一颜色的彩旗属组合问题，对于不同颜色的彩旗属排列问题．本例共计有 $M \times N$ 面彩旗，用尝试法编写程序时，需要 $M \times N$ 重循环，因 M 和 N 都是变量，所以属不定重循环问题．按不定重循环程序设计方法，其程序如下：

$M \times N$ 面彩旗排列程序和计算结果

```
#include "stdio.h"
```

```
#dcfine N 2
#define M 4

void main()
{
    int i, j, j0, k, k0, q, t=0, a[M*N+1], r[N+1]={0};
    int mm=0;
    for(i=1;i<=N;i++)
    {
        a[1]=i;
        r[i]=r[i]+1;
        k0=1;
        j0=2;
L1:;
        for(j=j0;j<=M*N;j++)
        {
L2:;
            for(k=k0;k<=N;k++)
            {
                if(r[k]==M)
                {
                    k0=k0+1;
                    goto L2;
                }
                else
                {
                    a[j]=k;
                    r[k]=r[k]+;
                    if(j<M*N)
                    {
                        j0=j+1;
                        k0=1;
                        goto L1;
                    }
                    else
                    {
                        for(q=1;q<=M*N;q++)
                            printf("%d",a[q]);
                        printf("\n");
                        t=t+1;
                        if(a[M*N-1]<a[M*N])
                        {
                            j0=M*N-1;
                            r[a[M*N]]=r[a[M*N]]-1;
                            r[a[M*N-1]]=r[a[M*N]-1]-1;
                            k0=a[M*N-1]+1;
                            goto L1;
                        }
```

```
                    else
                    {
                        j0=M*N-2;
                        while(a[j0]==N)
                            j0=j0-1;
                        for(q=M*N;q>=j0;q--)
                            r[a[q]]=r[a[q]]-1;
                        if(j0==1)
                            goto L3;
                        clsc
                        {
                            k0=a[j0]+1;
                            goto L1;
                        }
                    }
                }
            }
        }
        j0=j-1;
        r[a[j0]]=r[a[j0]]-1;
        if(j0==1)
            goto L3;
        k0=a[j0]+1;
        goto L1;
    }
L3:;
    }
    printf("\n%3d\n",t);
}
```

M(=4)*N*(=2)面彩旗排列表

11112222	11121222	11122122	11122212	11122221
11211222	11212122	11212212	11212221	11221122
11221212	11221221	11222112	11222121	11222211
12111222	12112122	12112212	12112221	12121122
12121212	12121221	12122112	12122121	12122211
12211122	12211212	12211221	12212112	12212121
12212211	12221112	12221121	12221211	12222111
21111222	21112122	21112212	21112221	21121122
21121212	21121221	21122112	21122121	21122211
21211122	21211212	21211221	21212112	21212121
21212211	21221112	21221121	21221211	21222111
22111122	22111212	22111221	22112112	22112121
22112211	22121112	22121121	22121211	22122111
22211112	22211121	22211211	22212111	22221111

注：表中 1 表示红旗，2 表示蓝旗；红旗、蓝旗各 4 面．*t* 为计数器，最后值为 70.

回溯法这一称呼虽雅，但和算法不吻合，不定重循环问题所用算法不是回溯法．回溯法实际上是倒推法．现举一个倒推法实例．

【例 2】有 12 个大小形状完全相同的球，其中有一个球的质量略为不同，请用天平量度三次，找出其中的异常球．

【解】本题的程序不少见，但给出真正的算法不多，本题所用算法为倒推法或回溯法．令质量不一致的球为非标准球，则

1．第三次使用天平时，天平左、右盘中最多只能有一个球属非标准球，即有效球个数各为 1．

2．第三次使用天平前最多只有三个球属非标准球，当有三个球可能是非标准球时，其中的两球曾在天平的一个盘上．

3．为了满足推理 2，当已知非标准球在 8 球之内时，第二次使用天平前必须做以下处理：

（1）适当利用已经确定的标准球；

（2）从天平左盘的球中取 2 个（或 1 个）出天平，从右盘的球中取 1 个（或 2 个）出天平；

（3）从天平左盘的球中取 1 个（或 2 个）放入右盘，从右盘的球中取 2 个（或 1 个）放入左盘；

当非标准球在 4 球之内时，算法简单，略．

4．第一次使用天平后，至少已知 4 球是标准球．

5．第一次使用天平时，天平左盘、右盘都必须只放 4 球．

虽然算法不是唯一的，但都必须按上面的推理进行．

对于尝试法最后一例，我们有意未给出带计算条件和计算过程的数学算式，只给出一个程序，程序中提及有限重循环加上转移语句模拟 n 重循环，其目的在于说明程序虽是算法，但仅是计算机使用的算法，它不能代替计算方法．倒推法是逻辑思维方法之一．像“华容道”、“九连环”这类题目，不用倒推法很难得到解．因倒推法不属计算方法，所以只顺便提及，也不给出程序．

11.3 递推算法

计算方法中，不少算法属于递推算法．递推算法分为一元递推算法、二元递推算法和广义递推算法三种．

11.3.1 一元递推算法

一般书中所介绍的递推算法指的都是一元递推算法．

【定义】凡能用算式

$$x_{k+p}=f(x_{k+1},x_{k+2},\cdots,x_{k+p-1}),\quad k=1,2,\cdots,n \tag{11.3-1}$$

表示的算法都称为一元递推算法．

显然，式（11.3-1）能够运算的前提是必须事先知道 $x_1, x_2, \cdots, x_{p-1}$．这些值称为表头值.

计算方法中，常微分方程数值算法中的所有显式格式都属于一元递推算法，所有一元递推算法所对应的基本程序结构都相同，都对应一个（一重）循环语句．为了更具体、更便于说明问题，下面将给出 4 阶 Runge-Kutta 算法的程序．4 阶 Runge-Kutta 算法为

$$\begin{cases} y_0 = a \\ k_1 = f(t_i, y_i) \\ k_2 = f\left(t_i + \dfrac{1}{2}h, y_i + \dfrac{1}{2}hk_1\right) \\ k_3 = f\left(t_i + \dfrac{1}{2}h, y_i + \dfrac{1}{2}hk_2\right) \\ k_4 = f(t_i + h, y_i + hk_3) \\ y_{i+1} = y_i + \dfrac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \end{cases} \tag{11.3-2}$$

下面是求解

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}t} = ty, & 0 \leqslant t \leqslant T = 1 \\ y(0) = 1 \end{cases} \tag{11.3-3}$$

的 4 阶 Runge-Kutta 程序，计算结果如表 11.1 所示：

```
double f(double t, double y)
{
    double z;
    z=t*y;
    return z;
}

#define M 10

void main()
{
    int i;
    double t[M+1], y[M+1], T, k1, k2, k3, k4, h, a=1;

    scanf("%lf",&T);
    h=T/M;
    for(i=0; i<=M; i++)
        t[i]=i*h;
    y[0]=a;
    for(i=0; i<M; i++)
    {
        k1=f(t[i],y[i]);
        k2=f(t[i]+0.5*h,y[i]+0.5*h*k1);
        k3=f(t[i]+0.5*h,y[i]+0.5*h*k2);
```

```
            k4=f(t[i]+h,y[i]+h*k3);
            y[i+1]=y[i]+h*(k1+2*k2+2*k3+k4)/6;
        }
        for(i=0; i<=M; i++)
            printf("%16.9lf",y[i]);
    }
```

表 11.1　4 阶 Runge-Kutta 法的计算结果（*T*=1）

t[*i*]	0.1	0.2	0.3	0.4	0.5
y[*i*]	1.005012521	1.020201340	1.046027859	1.083287065	1.133148446
t[*i*]	0.6	0.7	0.8	0.9	1.0
y[*i*]	1.197217347	1.277621279	1.377127695	1.499302362	1.648721007

4 阶 Runge-Kutta 算法属一元递推算法，其表头变量个数为 1，提供 y_0 后可按式（11.3-2）算出所有 y_i 的值．式（11.3-2）中，递推公式可归结为 $y_{i+1}=f(y_i)$．由于 f 的表达式较长，故先将 k_1,k_2,k_3,k_4 单独分开算，程序中 $h=T/N$ 属于简单算法，所有单步算法的程序都与 4 阶 Runge-Kutta 法的程序类似，仅有 $f(y_i)$ 不同，

对于多步算法，表头变量个数与步数一致，程序结构仍然类似．

11.3.2　二元递推算法

二元递推算法是这本教材首次归纳并提出的新算法类型，与一元递推算法一样，二元递推算式也由表头值和递推公式组成，只不过前者的表头值是几个变量，后者的表头值是几组变量．前者的递推公式中要包含一个下标，用一维数组表示，递推过程由下标变化过程决定；后者的递推公式包括两个下标，且与代数中的矩阵相联系，须用二维数组表示．计算方法中 Newton 差商表系数计算、有理插值中反差商表的反差商系数计算、Romberg 积分计算和杨辉三角系数计算，都可用二元递推算法表示．这里须指出的是，在归纳二元递推算法之前，必须先将所计算数据按矩阵方式排列．

计算方法中涉及的二元递推公式除了上述四个以外，Gauss 消元法、Gauss 列主元法、全主元法的消元算法等也属于二元递推算法．前面所指的四个二元递推算法是由前二行或前一列元素计算后一行或后一列元素的值，而 Gauss 消元法中却是由第 i 行元素的值计算第 $i+1$ 行到第 n 行元素的值，因此前者用二重循环语句实现递推过程，后者必须用三重循环语句．

鉴于二元递推公式有限，前面已经做过介绍，这里仅给出 Newton 差商表系数计算公式并附上有关解释．

计算公式为

$$\begin{cases} c_{i0}=y_i & i=0,1,\cdots,n \\ c_{ij}=\dfrac{c_{i+1,j-1}-c_{i,j-1}}{x_{i+j}-x_i} & j=1,2,\cdots,n;\quad i=0,1,\cdots,n-j \end{cases} \tag{11.3-4}$$

式（11.3-4）中，c_{i0} 为表头值，它等于 y_i，y_i 由输入语句给出，也可调用函数计算 $y_i = f(x_i)$．从式（11.3-4）可以看出，Newton 差商表系数是按列行顺序计算的，j 的值对应差商的阶．式（11.3-4）所对应的程序语句为

```
/*计算表头值*/
for(i=0l i<n; i++)
   c[i][0]=y[i];
/*计算系数值*/
for(j=1; j<=n; j++)
   for(i=0; i<=n-j; i++)
      c[i][j]=(c[i+1][j-1]-c[i][j-1])/(x[i+j]-x[i]);
```

这里必须指出，二元递推公式中各元素与矩阵元素对应，但不是说必须计算出整个矩阵．

给出二元递推公式的目的是，使算式与程序中语句的对应关系明确，从而降低程序设计的难度．现有计算方法教材中，除各 Gauss 消元法外，大多用图表 + 公式 + 实例描述算法，将图表 + 公式 +实例归纳成二元递推公式本身就是一种处理问题、解决问题的方法．归纳出用数学语言表述的带计算过程和计算条件的数学公式，对提高逻辑思维能力有帮助．

11.3.3　广义递推算法

广义递推算法包括累加和计算和连乘积计算，累加和计算和连乘积计算都属递推计算，都可用递推公式表示．

数学中用∑表示累加和，计算方法中数值积分部分的所有 Newton-Cotes 积分公式和 Gauss 积分公式都要计算累加和．Gauss 消元中的回代、改进平方根法中的三解分解也是计算累加和．用仿秦九韶法计算 Newton 插值多项式时，该公式也可认为是广义递推公式．可以将计算方法中的累加和计算公式分为

$$S = a_1 + a_1 + \cdots + a_k + \sum_{i=k+1}^{n} a_i \tag{11.3-5}$$

$$S = \sum_{i=1}^{n} a_i \tag{11.3-6}$$

上面两式中，a_i 称为加式通项，n 称为项数．式（11.3-5）中，$a_1, a_2, \cdots, a_k$ 项数有限，称为特殊项．

若用 S_j 表示前 j 项的和，显然有

$$S_{j+1} = S_j + a_{j+1} \tag{11.3-7}$$

式（11.3-7）是正宗的递推公式，所以我们将累加和计算称为广义递推计算，其算法称为广义递推算法．

式（11.3-5）对应的伪程序段的基本结构为

$$S = a_1 + a_2 + \cdots + a_k;$$

```
for(i=k+1; i<=n; i++)
{
```

```
    计算通项 a_i;
    S=S+a_i;
}
```

式（11.3-5）中，$a_1, a_2, \cdots, a_k$ 是已知的、确定的.

式（11.3-6）对应的伪程序段的基本结构为

```
S=0;
for(i=k+1; i<=n; i++)
{
    计算通项 a_i;
    S=S+a_i;
}
```

上面的两个程序中未使用数组 S[j]来表示 j 项和．原因如下：一是计算累加和时部分和无意义；二是累加和计算程序和算式都较简单，不必用数组，当然也可用数组元素表示 s_j .

下面是用 Simpson 公式计算 $\int_{-1}^{1}\frac{\mathrm{d}x}{1+x^2}$ 的计算公式，取 $n=4$：

$$S=\frac{h}{3}\left(f_0+f_n+2\sum_{i=1}^{n-1}f_{2i}+4\sum_{i=1}^{n}f_{2i-1}\right)$$

$$\begin{cases} f_i=f(a+ih) \\ h=(b-a)/(2n) \end{cases} \qquad (11.3\text{-}8)$$

所对应的程序语句为

```
double f(double x)
{
    double y;
    y=1/(1+x*x);
    return y;
}
#define  N  4
main()
{
    int i;
    double h, s, a, b, x;
    scanf("%lf%lf", &a, &b);
    h=(b-a)/(2*N);
    s=f(a)+f(b);
    for(i=1; i<N; i++)
    {
        x=a+2*i*h;
        s=s+2*f(x);
    }
    for(i=1; i<=N; i++)
    {
        x=a+(2*i-1)*h;
```

```
            s=s+4*f(x);
        }
        s=s*h/3;
        printf("%lf\n", s);
    }
```

本例的计算结果为 $s=1.487523$.

取 $h=\dfrac{b-a}{2n}$ 而不是 $\dfrac{b-a}{n}$ 的原因是，不让式（11.3-8）出现半节点，半节点与程序的对应关系不好，式（11.3-8）实际上是将 $f(a)+f(b)+2\sum_{i=1}^{n-1}f_{2i}$ 作为 $4\sum_{i=1}^{n}f_{2i-1}$ 的初值的累加和计算.

数学中连乘积计算所用符号为$\prod$，计算方法教材所含算法中只在 Lagrange 插值多项式的子项 $l_i(x)$ 和 Newton 插值多项式算式中用到$\prod$. Lagrange 插值主要用于定积分的数学公式推导，该算法用于插值时计算量大，稳定性不好，因而这里不做介绍. Newton 插值多项式的计算可以避开连乘用仿秦九韶法计算. 下面将 $N_n(x)\sum_{i=0}^{n}c_i\prod_{j=0}^{i-1}(x-x_j)$ 改写成

$$\begin{cases} S_n=c_{0n} \\ S_i=S_{i+1}(x-x_i)+c_{0i} \qquad i=n-1,n-2,\cdots,0 \end{cases} \tag{11.3-9}$$

此时，该算法成了递推算法，由于该算法实质上仍是计算累加和且部分和无意义，故可不用数组.

计算方法没有专门的连乘积计算. 这里不专门介绍连乘积计算.

11.4　迭代算法

迭代这个词在计算学科中用得很多，例如进行需求分析时，无论是通过用户和软件人员直接沟通，还是用原型法通过样机沟通，甚至利用业务基本平台进行需求分析，所有方法都是迭代法. 调试软件时使用的方法也是迭代法. 迭代法的最大特点是，在得到预期结果之前操作次数未知.

迭代法分为变量迭代法和向量迭代法两种.

11.4.1　变量迭代法

变量迭代算法是指用同一计算公式计算一个变量的值，直到前后两次计算结果之差不超过指定值（误差限）的算法.

所有变量迭代算法都可用下面的公式表示：

$$x^{(i+1)}=g(x^{(i)}),\ \ i=0,1,\cdots;\quad \left|x^{(i+1)}-x^{(i)}\right|>\varepsilon \tag{11.4-1}$$

当 $\left|x^{(i+1)}-x^{(i)}\right|\leqslant\varepsilon$ 时，迭代结束.

第 3 章所介绍的所有方法都属变量迭代法，常微分方程数值解法的所有隐式算法也属于

变量迭代法.

一元递推算法对应一个 for 型循环，迭代算法对应 while 型循环．式（11.4-1）所对应的伪程序段（令 $s=|x-x_0|$）如下：

```
给出 s 的先导值;
s=2*ep;
给出初值 x0;
while(s>ep)
{
    x=g(x0);
    s=fabs(x-x0);
    x0=x;
}
```

书中解方程的所有算法都属迭代法，迭代结束条件是 $|x^{(n+1)}-x^{(n)}|\leqslant\varepsilon$ 或 $|f(x^{(n+1)})|\leqslant\varepsilon$．但也可只用一个条件，且更有一般性，用迭代法解方程时，通常先找迭代结束条件，即 NOT（结束条件）= 迭代条件或循环条件.

在迭代法的基本程序模块中，while 语句前的语句为循环先导语句，该语句的目的是使循环语句起作用，先导语句中 s 的值必须满足循环条件，while 循环的循环体由迭代语句、数据传递语句及改变循环条件的有关语句组成．为了更方便地说明迭代法程序结构与语句之间的对应关系，下面直接给出改进 Euler 法的计算公式及程序．设

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}t}=ty, & 0\leqslant t\leqslant 1\\ y(0)=1\end{cases}$$

计算公式为

$$\begin{cases}y_{i+1}^{(m+1)}=y_i+\dfrac{h}{2}(f(t_i,y_i)+f(t_{i+1},y_{i+1}^{(m+1)}))\\ y_{i+1}^{(0)}=y_i+hf(t_i,y_i)\end{cases},\quad i=1,2,\cdots,N;\quad m=1,2,\cdots;$$

$$|y_{i+1}^{(m+1)}-y_{i+1}^{(m)}|>\varepsilon$$

所对应的程序语句为

```
double  f(double x, double y)
{
    double z;
    z=x*y;
    return z;
}
#define N  10
void main()
{
    int i;
    double y[N+1], t[N+1], T, y0, ep, s, h;
```

```
    scanf("%lf", &T);
    scanf("%lf", &y[0]);
    scanf("%lf", &ep);
    h=T/N;
    for(i=0; i<=N; i++)
        t[i]=i*h;
    for(i=1; i<=N; i++)
    {
        s=2*ep;
        y0=y[i-1]+h*f(t[i-1],y[i-1]);
        while(s>=ep);
        {
            y[i]=y[i-1]+0.5*h*(f(t[i-1],y[i-1])+f(t[i],y0));
            s=fabs(y[i]-y0);
            y0=y[i];
        }
    }
    for(i=1; i<=N; i++)
        printf("%lf\n", y[i]);
    getch();
}
```

计算结果如表 11.2 所示.

表 11.2　改进 Euler 法计算结果（T=1，ep=le−5）

$t[i]$	0.1	0.2	0.3	0.4	0.5
$y[i]$	1.005025	1.020253	1.046147	1.083510	1.133518
$t[i]$	0.6	0.7	0.8	0.9	1.0
$y[i]$	1.197789	1.278469	1.378350	1.501030	1.651133

改进 Euler 法要算的是 N 个 y_i ($i=1,2,\cdots,N$)，计算每一个 y_i 用的都是迭代法，取 Euler 法的计算结果为改进 Euler 法的初值，套用前面的基本程序结构便编出了上面的程序.

解常微分方程的所有隐式法的每一步都需要迭代.

11.4.2　向量迭代法

向量迭代法是指通过一个迭代公式可以解出一组变量或一个向量的计算方法．第 4 章中的所有算法及第 5 章的幂法、逆幂法都属于向量迭代算法.

向量迭代算法也对应一个 while 型循环，鉴于需要求解的是一个向量，遍历一个向量需要一重循环，迭代算式涉及另一个向量，遍历这个向量也需要一重循环，由于向量维数是一定数，故外循环为 for 型循环.

向量迭代法对应的语句留在数学实验中完成．第 4 章中给出了一个完整的程序，可供借

鉴．须再次说明的是，这里对算法的分类是按程序结构分类的，而不是按数学分类的．例如，

$$s=\sum_{i=0}^{\infty}\frac{1}{i!},\quad \frac{1}{i!}>\varepsilon \tag{11.4-2}$$

不属于累加和算法，而属于迭代法，因为该式对应的程序段是 while 型循环，而不是 for 型循环．

还须说明的是，在 C 语言中，for 型循环的功能极强，它可代替 while 型循环，这里只用 C 语言中 for 型循环的同步长有对应关系的这一子功能，该功能与 ALGOL 语言和 PASCAL 语言的 for 型循环的功能一致，目的在于使算法和程序段之间的对应关系明确，且程序中不含转移语句．

11.5 数学实验

本教材对少数算法给出了对应的 C 程序段，原因是现在已有不少流行的数学软件，这些软件中都包含了书中绝大多数算法的完整程序．

鉴于计算方法课时通常规定为 70 学时左右，在 70 学时内学完本教材前 10 章内容已不容易，因此我们对此不做硬性规定，而且也不主张所有学习计算方法的学生都要编写与各算法对应的某种语言的程序．但是，我们主张在讲解书中各章节之后让学生做数学实验，其目的在于提高学生的动手能力和使用数学软件的能力，便于学生熟悉各算法的功能、特点．

对于到底做什么实验和怎样做实验，本书不做具体规定，只给出下述建议．

（1）对于计算学科中各专业的学生或计算与信息专业的学生，每章结束后安排一个实验．

（2）做实验之前，学生在课余时间编写程序，计算学科中各专业的学生大多数拥有自己的计算机，做到这一点不难．正式实验时主要是在老师的指导下调试程序和进行算法分析．应在实验前选好程序算例．

（3）每次实验后必须书写实验报告．

普通物理、普通化学、材料力学等学科的实验目的多重于温故，实验内容、器材、实验结果都已预先知道．本门学科的实验温故只是目的之一，知新更为重要．由于学生所选程序用例不同，结果也会不一样．当然，即使我们在书中给出了算法和对应的程序段，也不能保证学生所编写的程序相同，因此对教师的要求也相对更高．

由于计算方法中的实验还未列在教学大纲之内，而且以往没有实验，所以需要在实践中完善．

参 考 文 献

［1］冯康．数值计算方法[M]．北京：国防工业出版社，1978．

［2］林成森．数值计算方法（上、下）[M]．北京：科学出版社，2005．

［3］（美）霍恩贝克（Robert W. Hornbeck）著，刘元久译．数值方法[M]．北京：中国铁道出版社，1982．

［4］李尚志．数学实验[M]．北京：高等教育出版社，1999．

［5］王仁宏．数值逼近[M]．北京：高等教育出版社，1999．

［6］张世禄，陈豫眉．计算方法[M]．北京：电子工业出版社，2006．

［7］J. Storer, R. Bulirsch．Introduction to Numerical Analysis, Second Edition [M]．广州：世界图书出版社，1998．

［8］Richard L. Burden, J. Dulas Faires．Numerical Analysis, Seventh Edition[M]．北京：高等教育出版社，2003．

［9］David Kincaid, Ward Cheney．Numerical Analysis[M]．北京：机械工业出版社，2003．

［10］张海藩．软件工程（第三版）[M]．北京：清华大学出版社，1998．

［11］B. W. Kernighan, P. J. Plauger 著，晏晓焰译．程序设计技巧[M]．北京：清华大学出版社，1985．

［12］张世禄，陈毅清．信息学竞赛程序设计方法[M]．北京：电子工业出版社，2007．